

PIC18F1230/1330
数据手册

采用纳瓦技术
配备高性能 PWM 和 A/D 的
18/20/28 引脚
增强型闪存单片机

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 "牢不可破" 的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe、位于俄勒冈州 Gresham 及位于加利福尼亚州 Mountain View 的全球总部、设计中心和晶圆生产厂均通过了 ISO/TS-16949:2002 认证。公司在 PIC® 8 位单片机、 KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

MICROCHIP

PIC18F1230/1330

采用纳瓦技术配备高性能 PWM 和 A/D 的 18/20/28 引脚增强型闪存单片机

14 位电源控制 PWM 模块：

- 多达 6 路 PWM 输出
  - 互补或独立输出
- 边沿或中心对齐的工作模式
- 灵活的死区发生器
- 硬件故障保护输入
- 占空比和周期同步更新
  - 灵活的特殊事件触发信号输出

灵活的振荡器结构：

- 4 种晶振模式，频率最高可达 40 MHz
- 4 倍频锁相环 （Phase Lock Loop， PLL）—— 可供晶振和内部振荡器使用
- 两种外部 RC 模式，频率最高可达 4 MHz
- 两种外部时钟模式，频率最高可达 40 MHz
- 内部振荡器电路：
  - 8 种用户可选择的频率，范围从 31 kHz 至 8 MHz
  - 与 PLL 结合使用时提供更宽的时钟范围，从 31 kHz 至 32 MHz
  - 用户可对频率进行调整来补偿频率漂移
- 使用 Timer1 （工作频率为 32 kHz）的辅助振荡器
- 故障保护时钟监视器：
  - 当外设时钟停止时可使器件安全关闭

功耗管理模式：

- 运行：CPU 工作，外设工作
- 空闲：CPU 停止，外设工作
- 休眠：CPU 停止，外设停止
- 空闲模式下的电流消耗可降至 5.8 μA （典型值）
- 休眠模式下的电流消耗可降至 0.1 μA （典型值）
- Timer1 振荡器的电流消耗：在 32 kHz、 2V 条件下为 1.8 μA （典型值）
- 看门狗定时器 （Watchdog Timer， WDT）的电流消耗：2.1 μA （典型值）
- 振荡器的双速启动

外设特点：

- 高灌 / 拉电流 25 mA/25 mA
- 多达 4 个可编程外部中断
- 4 个输入电平变化中断
- 增强型可寻址 USART 模块：
  - 支持 RS-485、 RS-232 和 LIN 1.2
  - 使用内部振荡器电路的 RS-232 操作 （无需外部晶振）
  - 遇到起始位时自动唤醒
  - 自动波特率检测
- 多达 4 路通道的 10 位模数转换器（Analog-to-Digital Converter， ADC）模块：
  - 自动采集功能
  - 可在休眠模式下进行转换
- 多达 3 个模拟比较器
- 可编程的比较器参考电压
- 可编程的 15 级低电压检测 （Low-Voltage Detection， LVD）模块：
  - 支持在检测到低电压时产生中断

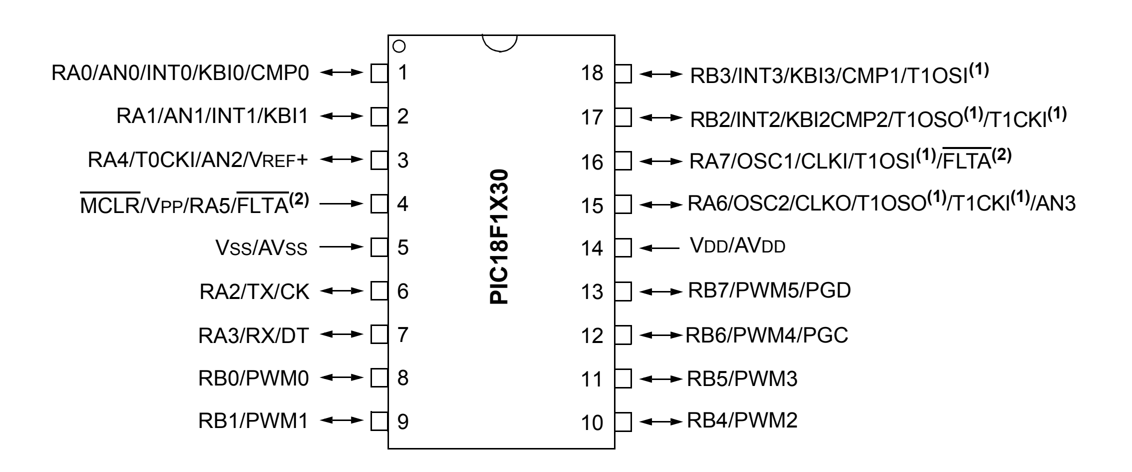
单片机的特殊性能：

- 优化的 C 编译器架构，带有可选的扩展指令集
- 闪存数据保存时间：大于 40 年
- 可在软件控制下自编程
- 中断优先级
- 8 x 8 单周期硬件乘法器
- 扩展的看门狗定时器 （WDT）：
  - 可编程周期从 4 ms 至 131s
- 可编程代码保护
- 通过两个引脚进行单电源在线串行编程（In-Circuit Serial Programming™， ICSP™）
- 通过两引脚进行在线调试 （In-Circuit Debug， ICD）
- 宽工作电压范围 （2.0V 至 5.5V）

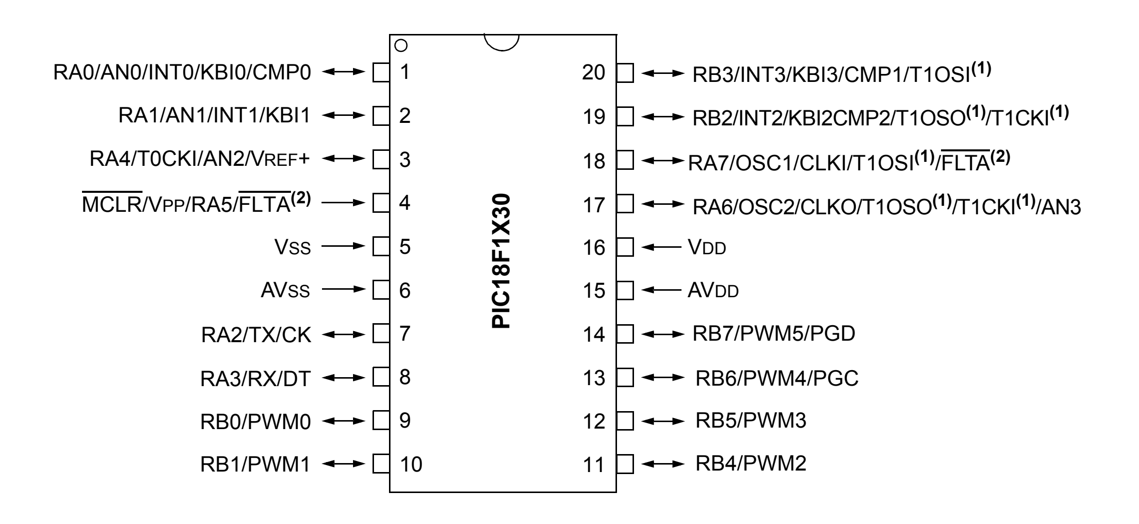
器件	程序存储器		数据存储器		I/O	10 位 ADC 通道	EUSART	模拟比较器	14 位 PWM （通道数）	16 位定时器
	闪存 （字节）	# 单字指令	SRAM （字节）	EEPROM （字节）						
PIC18F1230	4096	2048	256	128	13	4	有	3	6	2
PIC18F1330	8192	4096	256	128	13	4	有	3	6	2

引脚图

18 引脚 PDIP 和 SOIC



20 引脚 SSOP

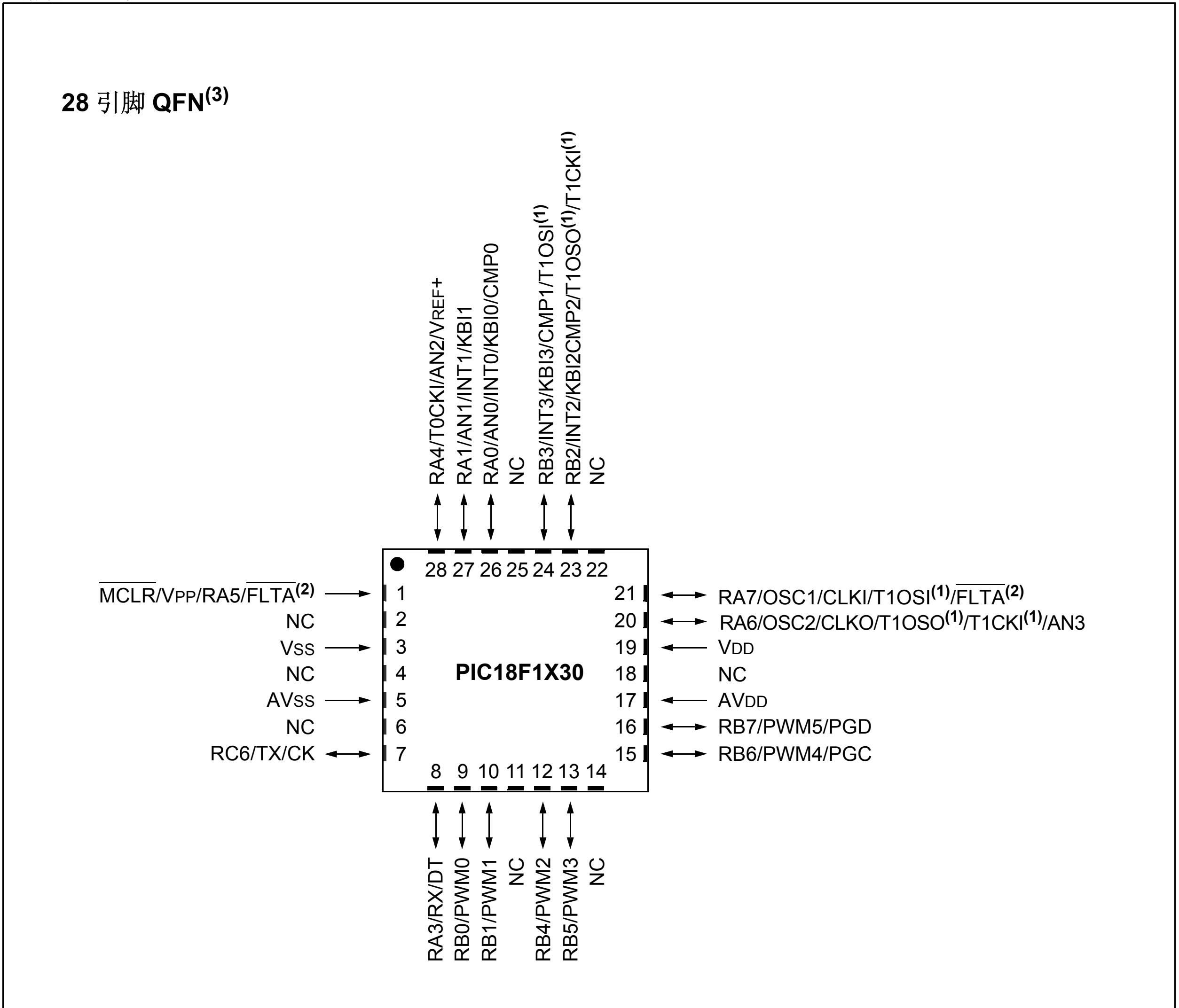


注 **1：** T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。
2： FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。

 超前信息

引脚图 （续）

28 引脚 QFN(3)



注 1： T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。
2： FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。
3： 建议用户将此类器件外封装的中心金属衬底接地。

目录

致客户

我们旨在提供最佳文档供客户正确使用 Microchip 产品。为此，我们将不断改进出版物的内容和质量，使之更好地满足您的要求。出版物的质量将随新文档及更新版本的推出而得到提升。

如果您对本出版物有任何问题和建议，请通过电子邮件联系我公司 TRC 经理，电子邮件地址为 **CTRC@microchip.com**，或将本数据手册后附的 **《读者反馈表》**传真到 86-21-5407 5066。我们期待您的反馈。

最新数据手册

欲获得本数据手册的最新版本，请查询我公司的网站：

http://www.microchip.com

查看数据手册中任意一页下边角处的文献编号即可确定其版本。文献编号中数字串后的字母是版本号，例如：DS30000A是DS30000 的 A 版本。

勘误表

现有器件可能带有一份勘误表，描述了实际运行与数据手册中记载内容之间存在的细微差异以及建议的变通方法。一旦我们了解到器件 / 文档存在某些差异时，就会发布勘误表。勘误表将注明其所适用的硅片版本和文件版本。

欲了解某一器件是否存在勘误表，请通过以下方式之一查询：

- Microchip 网站：http://www.microchip.com
- 当地 Microchip 销售办事处 （见最后一页）

在联络销售办事处时，请说明您所使用的器件型号、硅片版本和数据手册版本 （包括文献编号）。

客户通知系统

欲及时获知 Microchip 产品的最新信息，请到我公司网站 **www.microchip.com** 上注册。

注：

1.0 器件概述

该文档包含针对以下器件的信息：

- PIC18F1230
- PIC18F1330

该系列具备所有 PIC18 单片机固有的优点——即，以优惠的价格提供出色的计算性能，此外还具有高耐久性的增强型闪存程序存储器。除此之外，PIC18F1230/1330 系列还引进了增强型设计，使得该系列单片机成为许多高性能以及电源和电机控制应用的明智选择。

外设特性包括：

- 带有可编程死区时间的 14 位高分辨率电源控制 PWM 模块 （Power Control PWM Module，PCPWM）

PCPWM能够产生多达6路可插入死区时间的互补PWM 输出。由片外模拟比较器或数字故障输入引脚（FLTA）检测驱动的电流是否超过规范值。

PIC18F1230/1330 器件还具有闪存程序存储器和内部 RC 振荡器。

1.1 新的内核特性

1.1.1 纳瓦技术

PIC18F1230/1330 系列的所有器件具有一系列能在工作时显著降低功耗的功能。主要包括以下几项：

- **备用运行模式**：通过将 Timer1 或内部振荡器电路作为单片机时钟源，可使代码执行时的功耗降低 90%。
- **多种空闲模式**：单片机还可工作在其 CPU 内核禁止而外设仍然运行的情况下。处于这些状态时，功耗能降得更低，只有正常工作时的 4%。
- **动态模式切换**：在器件工作期间可由用户代码调用功耗管理模式，允许用户将节能的理念融入到他们的应用软件设计中。
- **关键模块的低功耗**：Timer1 和看门狗定时器的功耗需求被降到最低程度。请参见**第 22.0 节 “电气规范”**了解具体数值。

1.1.2 多个振荡器选项和特性

PIC18F1230/1330 系列的所有器件提供 10 种振荡器选项，使用户在开发应用硬件时有很大的选择范围。这些选项包括 ：

- 四种晶振模式，使用晶振或陶瓷谐振器。
- 两种外部时钟模式，提供使用两个引脚 （振荡器输入引脚和四分频时钟输出引脚）或一个引脚 （振荡器输入引脚，四分频时钟输出引脚重新分配为通用 I/O 引脚）的选项。
- 两种外部 RC 振荡器模式，具有与外部时钟模式相同的引脚选项。
- 一个内部振荡器模块 （提供一个 8 MHz 的时钟源），和一个 INTRC 时钟源 （振荡频率大约为 31 kHz），以及 6 种用户可选择的时钟频率 （从 125 kHz 到 4 MHz）。因此共有 8 种时钟频率可供选择。此选项可以将两个振荡器引脚作为额外的通用 I/O 引脚。
- 一个锁相环 （PLL）倍频器，可用于高速晶振和内部振荡模式，使时钟速度可高达 40 MHz。 PLL 和内部振荡器配合使用，可以向用户提供频率范围从 31 kHz 至 32 MHz 的时钟，而且不需要使用外部晶振或时钟电路。

除了可被用作时钟源，内部振荡器电路还提供了一个稳定的参考源，增加了以下功能以使器件更安全地工作：

- **故障保护时钟监视**：该功能部件不停地监视主时钟源，将其与内部振荡器提供的参考信号作比较。如果主时钟发生了故障，单片机会将时钟源切换到内部振荡器电路，使器件可继续低速工作或安全地关机。
- **双速启动**：该功能允许在上电复位或从休眠模式唤醒时将内部振荡器用作时钟源，直到主时钟源可用为止。

1.2 其他特性

- **存储器耐久性：**程序存储器和数据 EEPROM 的增强型闪存单元经评测，能经受数千次擦 / 写，程序存储器高达 100,000 次， EEPROM 高达 1,000,000 次。如果不刷新，数据保存期保守地估计在 40 年以上。
- **自编程性：**这些器件能在内嵌软件控制下对各自的程序存储空间进行写操作。通过使用位于受保护的引导区 （程序存储器顶端）中的自举程序，应用程序在现场可实现自我更新。
- **扩展指令集：**PIC18F1230/1330 系列在 PIC18 指令集的基础上进行了扩展，添加了 8 条新指令和变址寻址模式。此扩展可以使用一个器件配置选项使能，它是为优化重入代码而特别设计的，这些代码是使用高级语言 （如 C 语言）开发的。
- **电源控制 PWM 模块：**该模式最多可提供 6 路调制输出来控制半桥和全桥驱动器。其他功能包括检测到故障时的自动断电和自动重启 （自动重启能在故障条件被清除时再次激活输出）。
- **增强型可寻址 USART：**此串行通信模块可进行标准的 RS-232 通信并支持 LIN 总线协议。其他增强的功能包括自动波特率检测和分辨率更高的 16 位波特率发生器。当单片机使用内部振荡器电路时， EUSART 为与外界通信的应用程序提供稳定的通信方式，而不需要使用外部晶振也无需额外的功耗。
- **10 位 A/D 转换器：**该模块带有可编程采集时间，从而不必在选择通道和启动转换之间等待一个采样周期，因而减少了代码开销。
- **扩展的看门狗定时器 （WDT）：**该增强的看门狗定时器添加了 16 位预分频器，扩展了超时周期范围，并在整个工作电压和温度范围内保持稳定。请参见**第 22.0 节 "电气规范"**了解超时周期的具体数值。

1.3 系列中各产品的具体信息

PIC18F1230/1330 系列中的器件具有 18 引脚、20 引脚和 28 引脚三种封装形式。

这些器件在以下方面有所不同：

1. 闪存程序存储器 （PIC18F1230 器件为 4 KB， PIC18F1330 器件为 8 KB）。

该系列器件的其他功能都是相同的。表 1-1 汇总了这些功能。

图 1-1 给出了 PIC18F1220/1320 器件架构的框图。表 1-2 给出了该系列器件的引脚配置。

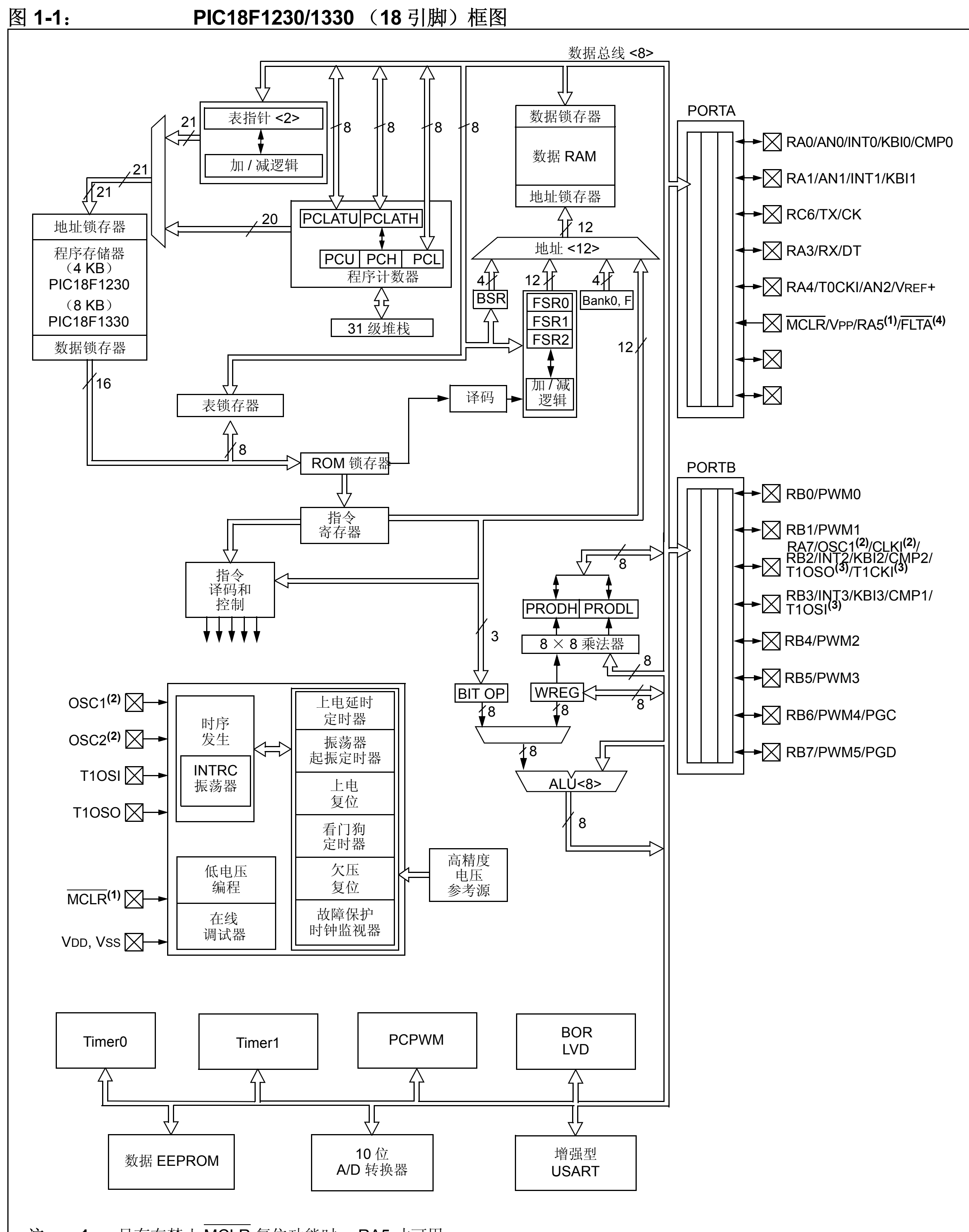
和所有 Microchip PIC18 器件一样，PIC18F1230/1330 系列也有标准器件和低压器件可供选择。器件编号中标有字母 "F"的是带有增强型闪存存储器的标准器件 （如 PIC18**F**1330），其工作电压 VDD 范围为 4.2V 至 5.5V。编号中标有 "LF"的低压器件（如 PIC18**LF**1330）可工作在扩展的 VDD 范围 （2.0V 至 5.5V）中。

超前信息

表 **1-1**： 器件特性

特性	PIC18F1230	PIC18F1330
工作频率	DC — 40 MHz	DC — 40 MHz
程序存储器 （字节）	4096	8192
程序存储器 （指令）	2048	4096
数据存储器 （字节）	256	256
数据 EEPROM 存储器 （字节）	128	128
中断源	17	17
I/O 端口	端口 A 和 B	端口 A 和 B
定时器	2	2
电源控制 PWM 模块	6 通道	6 通道
串行通信	增强型 USART	增强型 USART
10 位模数转换模块	4 个输入通道	4 个输入通道
复位 （和延时）	POR、 BOR、 RESET 指令、 堆栈满、 堆栈下溢 （PWRT 和 OST）、 MCLR （可选）和 WDT	POR、 BOR、 RESET 指令、 堆栈满、 堆栈下溢 （PWRT 和 OST）、 MCLR （可选）和 WDT
可编程低电压检测	有	有
可编程欠压复位	有	有
指令集	75 条指令； 启用了扩展指令集后 总共为 83 条指令	75 条指令； 启用了扩展指令集后 总共为 83 条指令
封装	18 引脚 PDIP 18 引脚 SOIC 20 引脚 SSOP 28 引脚 QFN	18 引脚 PDIP 18 引脚 SOIC 20 引脚 SSOP 28 引脚 QFN

图 1-1： PIC18F1230/1330 （18 引脚）框图



注 1： 只有在禁止 MCLR 复位功能时， RA5 才可用。
2： OSC1、 OSC2、CLKI 和 CLKO 只有在选择振荡器模式时才可用，并且这些引脚不被用作数字 I/O 引脚。更多信息，请参见**第 2.0 节 “振荡器配置”**。
3： T1OSI 和 T1OSO/T1CKI 的引脚位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。
4： FLTA 的引脚位置取决于 CONFIG3H 中 FLTAMX 配置位的值。

 超前信息

表 **1-2：** **PIC18F1230/1330 I/O** 引脚配置说明

引脚名称	引脚号			引脚类型	缓冲器类型	说明
	PDIP, SOIC	**SSOP**	**QFN**			
$\overline{MCLR}$/VPP/RA5/$\overline{FLTA}$	4	4	1			主清零 （输入）、编程电压 （输入）或故障检测输入。
$\overline{MCLR}$				I	ST	主清零 （复位）输入。此引脚为低电平时器件复位。
VPP				I	模拟	编程电压输入。
RA5				I	ST	数字输入。
$\overline{FLTA}$(1)				I	ST	PWM 故障检测输入
RA7/OSC1/CLKI/T1OSI/FLTA	16	18	21			振荡器晶振、外部时钟输入、 Timer1 振荡器输入或故障检测输入。
RA7				I/O	ST	数字 I/O。
OSC1				I	模拟	振荡器晶振输入或外部时钟源输入。
CLKI				I	模拟	外部时钟源输入。
T1OSI(2)				I	模拟	Timer1 振荡器输入。
$\overline{FLTA}$(1)				I	ST	PWM 故障检测输入。
RA6/OSC2/CLKO/T1OSO/T1CKI/AN3	15	17	20			振荡器晶振、时钟输出、 Timer1 振荡器输出或模拟输入。
RA6				I/O	ST	数字 I/O。
OSC2				O	模拟	振荡器晶振输出或外部时钟源输入。
CLKO				O	模拟	外部时钟源输出。
T1OSO(2)				O	模拟	Timer1 振荡器输出。
TICKI(2)				I	ST	Timer1 时钟输入。
AN3				I	模拟	模拟输入 3。

图注： TTL = TTL 兼容输入　　CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入　　I = 输入
O = 输出　　P = 电源

注 **1：** FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。
2： T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

表 1-2: PIC18F1230/1330 I/O 引脚配置说明 （续）

引脚名称	引脚号 PDIP, SOIC	引脚号 SSOP	引脚号 QFN	引脚类型	缓冲器类型	说明
						PORTA 是双向 I/O 端口。
RA0/AN0/INT0/KBI0/CMP0	1	1	26			
RA0				I/O	TTL	数字 I/O。
AN0				I	模拟	模拟输入 0。
INT0				I	ST	外部中断 0。
KBI0				I	TTL	电平变化中断引脚。
CMP0				I	模拟	比较器 0 输入。
RA1/AN1/INT1/KBI1	2	2	27			
RA1				I/O	TTL	数字 I/O。
AN1				I	模拟	模拟输入 1。
INT1				I	ST	外部中断 1。
KBI1				I	TTL	电平变化中断引脚。
RA2/TX/CK	6	7	7			
RA2				I/O	TTL	数字 I/O。
TX				O	—	EUSART 异步发送。
CK				I/O	ST	EUSART 同步时钟。
RA3/RX/DT	7	8	8			
RA3				I/O	TTL	数字 I/O。
RX				I	ST	EUSART 异步接收。
DT				I/O	ST	EUSART 同步数据。
RA4/T0CKI/AN2/VREF+	3	3	28			
RA4				I/O	TTL	数字 I/O。
T0CKI				I	ST	Timer0 外部时钟输入。
AN2				I	模拟	模拟输入 2。
VREF+				I	模拟	A/D 参考电压 （高电平端）输入。

图注： TTL = TTL 兼容输入　　CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入　　I = 输入
O = 输出　　P = 电源

注 1： FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。
2： T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

表 **1-2**： **PIC18F1230/1330 I/O** 引脚配置说明 （续）

引脚名称	引脚号			引脚类型	缓冲器类型	说明
	PDIP, SOIC	**SSOP**	**QFN**			
						PORTB 是双向 I/O 端口。
RB0/PWM0	8	9	9			
RB0				I/O	TTL	数字 I/O。
PWM0				O	—	PWM 模块输出 PWM0。
RB1/PWM1	9	10	10			
RB1				I/O	TTL	数字 I/O。
PWM1				O	—	PWM 模块输出 PWM1。
RB2/INT2/KBI2/CMP2/ T1OSO/T1CKI	17	19	23			
RB2				I/O	TTL	数字 I/O。
INT2				I	ST	外部中断 2。
KBI2				I	TTL	电平变化中断引脚。
CMP2				I	模拟	比较器 2 输入。
T1OSO**(2)**				O	模拟	Timer1 振荡器输出。
T1CKI**(2)**				I	ST	Timer1 时钟输入。
RB3/INT3/KBI3/CMP1/ T1OSI	18	20	24			
RB3				I/O	TTL	数字 I/O。
INT3				I	ST	外部中断 3。
KBI3				I	TTL	电平变化中断引脚。
CMP1				I	模拟	比较器 1 输入。
T1OSI**(2)**				I	模拟	Timer1 振荡器输入。
RB4/PWM2	10	11	12			
RB4				I/O	TTL	数字 I/O。
PWM2				O	—	PWM 模块输出 PWM2。
RB5/PWM3	11	12	13			
RB5				I/O	TTL	数字 I/O。
PWM3				O	—	PWM 模块输出 PWM3。
RB6/PWM4/PGC	12	13	15			
RB6				I/O	TTL	数字 I/O。
PWM4				O	—	PWM 模块输出 PWM4。
PGC				I	ST	在线调试器和 ICSP™ 编程时钟引脚。
RB7/PWM5/PGD	13	14	16			
RB7				I/O	TTL	数字 I/O。
PWM5				O	—	PWM 模块输出 PWM5。
PGD				O	—	在线调试器和 ICSP 编程数据引脚。

图注： TTL = TTL 兼容输入　　CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入　　I = 输入
O = 输出　　P = 电源

注 **1**： FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。

2： T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

表 1-2： PIC18F1230/1330 I/O 引脚配置说明 （续）

引脚名称	引脚号			引脚类型	缓冲器类型	说明
	PDIP, SOIC	SSOP	QFN			
VSS	5	5	3	P	—	逻辑电路和 I/O 引脚的参考地。
VDD	14	16	19	P	—	逻辑电路和 I/O 引脚的正电源。
AVSS	5	6	5	P	—	A/D 转换模块的参考地。
AVDD	14	15	17	P	—	A/D 转换模块的正电源。
NC	—	—	2, 4, 6, 11, 14, 18, 22, 25	—	—	无连接。

图注： TTL = TTL 兼容输入 CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入 I = 输入
O = 输出 P = 电源

注 1： FLTA 引脚的位置取决于 CONFIG3H 中 FLTAMX 配置位的值。
2： T1OSI 和 T1OSO/T1CKI 引脚的位置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

 超前信息

2.0 振荡器配置

2.1 振荡器类型

PIC18F1230/1330 器件可在 10 种不同的振荡器模式下工作。用户可通过设置配置寄存器 1H 中的配置位 FOSC3:FOSC0，选择这 10 种模式中的一种：

1. LP 低功耗晶振
2. XT 晶振 / 谐振器
3. HS 高速晶振 / 谐振器
4. HSPLL 使能 PLL 的高速晶振 / 谐振器
5. RC 外部阻容振荡器，在 RA6 引脚上输出 Fosc/4 时钟信号
6. RCIO 外部阻容振荡器，RA6 引脚作为 I/O 引脚
7. INTIO1 内部振荡器，在 RA6 引脚上输出 Fosc/4 时钟信号， RA7 引脚作为 I/O 引脚
8. INTIO2 内部振荡器，RA6 和 RA7 均作为 I/O 引脚
9. EC 外部时钟，输出为 Fosc/4 时钟信号
10. ECIO 外部时钟，RA6 作为 I/O 引脚

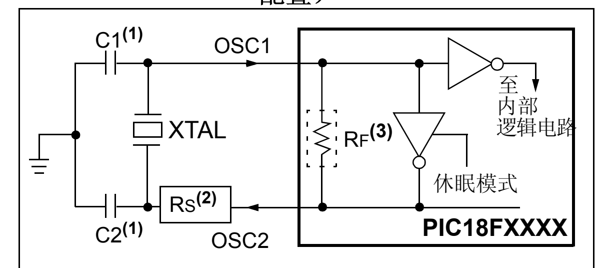
2.2 晶振 / 陶瓷谐振器

在 XT、LP、HS 或 HSPLL 振荡模式中，晶振或陶瓷谐振器与 OSC1 和 OSC2 引脚相连以建立振荡信号。图 2-1 显示了引脚连接方式。

振荡器的设计要求使用平行切割的晶体。

注：	使用顺序切割的晶体，会使振荡器产生的频率不在晶体制造厂商所给的参数范围内。

图 2-1： 晶振 / 陶瓷谐振器工作原理（XT、 LP、 HS 或 HSPLL 配置）



注 1： 如需了解 C1 和 C2 的初始值，请参见表 2-1 和表 2-2。
2： AT 条形切割的晶体可能会需要一个串联电阻（RS）。
3： RF 的值根据所选择的振荡器模式而定。

表 2-1： 陶瓷谐振器的电容选择

使用的典型电容值：			
模式	频率	OSC1	OSC2
XT	3.58 MHz 4.19 MHz 4 MHz 4 MHz	15 pF 15 pF 30 pF 50 pF	15 pF 15 pF 30 pF 50 pF

上述电容值仅供设计参考。

要得到合适的振荡器工作状况，可能需要不同的电容值。用户应在应用的预期 VDD 和温度范围内测试振荡器的性能。

欲知更多信息，请参见表 2-2 后的 “注”。

表 2-2： 晶振的电容选择

振荡器类型	晶振频率	已测试的典型电容值：	
		C1	**C2**
LP	32 kHz	30 pF	30 pF
XT	1 MHz 4 MHz	15 pF 15 pF	15 pF 15 pF
HS	4 MHz 10 MHz 20 MHz 25 MHz	15 pF 15 pF 15 pF 15 pF	15 pF 15 pF 15 pF 15 pF

上述电容值仅供设计参考。

要得到合适的振荡器工作状况，可能需要不同的电容值。用户应在应用的预期 V_{DD} 和温度范围内测试振荡器的性能。

欲知更多信息，请参见本表后的 “注”。

注 1：较高的电容值可以提高振荡器的稳定性，但同时也会延长起振时间。

2：当工作电压 V_{DD} 低于 3V，或使用某些陶瓷谐振器时，可能需要使用 HS 振荡器模式或改用晶振。

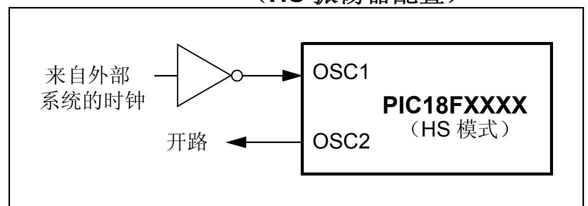
3：因为每种谐振器 / 晶振都有其自身特性，用户应当向谐振器 / 晶振制造厂商询问外部元件的相应值。

4：为避免对低驱动电平规格的晶体造成过驱动，可能会需要使用电阻 R_S。

5：请在应用中的预期 V_{DD} 和温度范围内验证振荡器的性能。

如图 2-2 所示，在 HS 模式下，外部时钟源也可以连接在 OSC1 引脚。

图 2-2： 外部时钟输入工作原理（**HS** 振荡器配置）

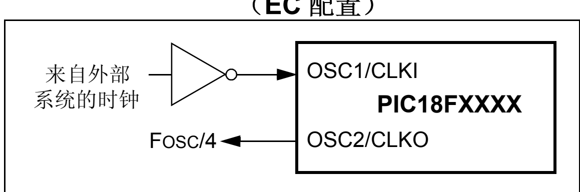


2.3 外部时钟输入

EC 和 ECIO 振荡模式要求外部时钟源连接到 OSC1 引脚。在上电复位或从休眠模式退出后，不需要振荡器起振延时。

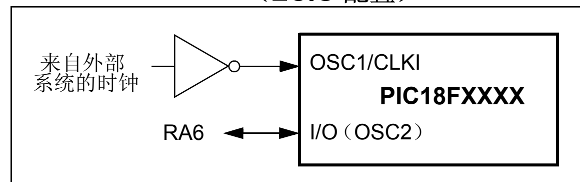
在 EC 振荡器模式下，可在 OSC2 引脚输出振荡器频率的 4 分频信号。此信号可用于测试或同步其他逻辑。图 2-3 显示了 EC 振荡器模式的引脚连接方式。

图 2-3： 外部时钟输入工作原理（**EC** 配置）



ECIO 振荡器模式的工作原理与 EC 模式类似，不同之处在于 OSC2 引脚变成了一个额外的通用 I/O 引脚。该 I/O 引脚成为 PORTA 的 bit 6 （RA6）。图 2-4 显示了 ECIO 振荡器模式的引脚连接方式。

图 2-4： 外部时钟输入工作原理（**ECIO** 配置）



 超前信息

2.4 RC 振荡器

对于对时序要求不高的应用，“RC”和 “RCIO”器件选项可以进一步节约成本。实际的振荡器频率随以下几个因素变化：

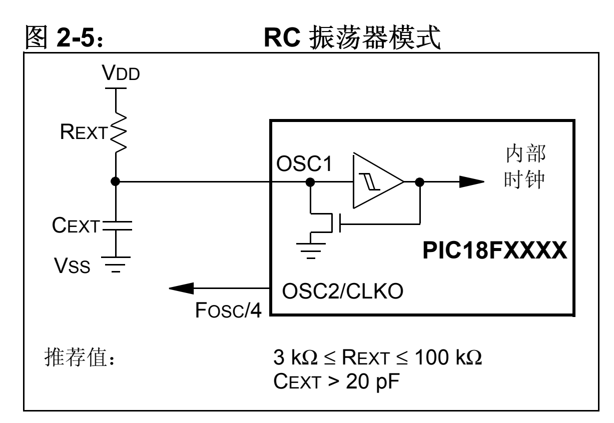
- 供电电压
- 外部电阻 （REXT）和电容 （CEXT）的值
- 工作温度

给定同样的器件、工作电压和温度以及元件值，器件频率仍然会各不相同。这些频率上的差异是由诸如以下的因素引起的：

- 正常制造工艺的差别
- 不同封装类型的引线电容差异 （尤其当 CEXT 值较小时）
- REXT 和 CEXT 在容限范围内的数值波动

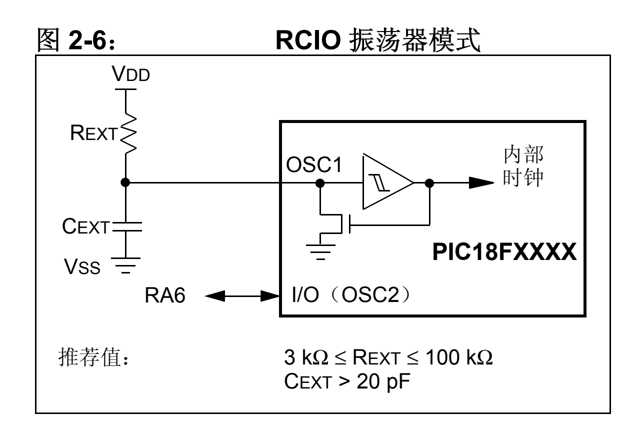
在 RC 振荡器模式下，振荡器频率的 4 分频信号可由 OSC2 引脚输出。此信号可用于测试或同步其他逻辑。图 2-5 显示了 R/C 组合电路的连接方式。

图 2-5： RC 振荡器模式



RCIO 振荡器模式 （图 2-6）的工作原理与 RC 模式类似，不同之处在于 OSC2 引脚变成了一个额外的通用 I/O 引脚。该 I/O 引脚成为 PORTA 的 bit 6 （RA6）。

图 2-6： RCIO 振荡器模式



2.5 PLL 倍频器

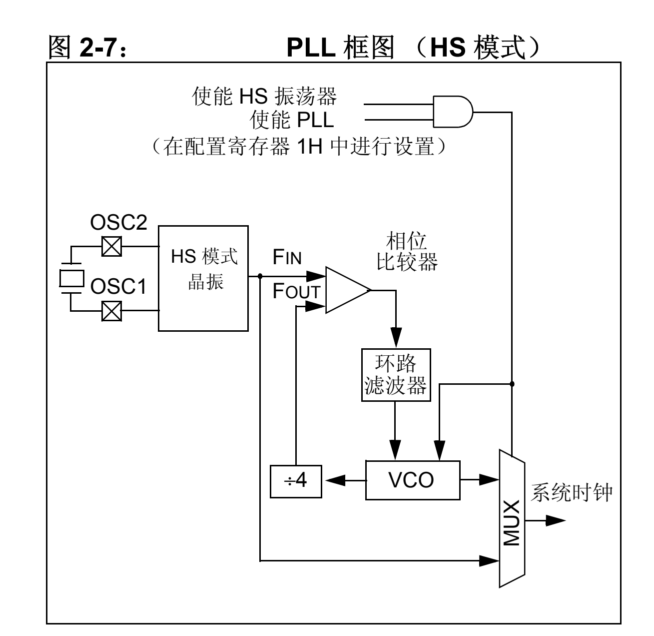
如果用户希望使用低频晶振电路或通过晶振将器件频率调节至其最高额定频率，可以选择使用锁相环 （PLL）电路。对于担心高频晶振引起 EMI 或需要内部振荡器提供高速时钟的用户而言，这样做可能会有用。

2.5.1 HSPLL 振荡器模式

HSPLL 模式使用 HS 模式振荡器产生最高 10 MHz 的频率。然后 PLL 将振荡器输出频率乘以 4，从而产生最高为 40 MHz 的内部时钟频率。在 HSPLL 振荡器模式下 PLLEN 位不可用。

仅当 FOSC3:FOSC0 配置位被编程为 HSPLL 模式（= `0110`）时，晶振才可使用 PLL。

图 2-7： PLL 框图 （HS 模式）



2.5.2 PLL 和 INTOSC

在选定的振荡器模式下，内部振荡器电路也可以使用 PLL。在此配置下， PLL 用软件使能并产生最高为 32 MHz 的时钟输出。第 **2.6.4 节 “INTOSC 模式下的 PLL”**说明了 INTOSC 在使用 PLL 时的工作原理。

2.6 内部振荡器电路

PIC18F1230/1330 器件包含一个内部振荡器电路，可产生两种均可充当单片机时钟源的时钟信号，从而避免了在 OSC1 和 / 或 OSC2 引脚上使用外部振荡器电路。

主输出（INTOSC）是一个 8 MHz 的时钟源，可以用于直接驱动器件时钟。它还可以驱动后分频器，提供 31 kHz 至 4 MHz 之间的时钟频率。选择 125 kHz 至 8 MHz 的时钟频率也就使能了 INTOSC 输出。

另一个时钟源是内部 RC 振荡器 （INTRC），它提供了标称的 31 kHz 输出。选择 INTRC 作为器件的时钟源，也就使能了内部 RC 振荡器；当使能以下任何一项时，也将自动使能内部 RC 振荡器：

- 上电延时定时器
- 故障保护时钟监视器
- 看门狗定时器
- 双速启动

在第 **19.0 节 “CPU 的特殊功能”**中对这些特性进行了更详细地讨论。

通过配置 OSCCON 寄存器的 IRCF 位，选择时钟源频率 （INTOSC 直接输出、 INTRC 直接输出或 INTOSC 后分频器的输出）（见第 22 页）。

2.6.1 INTIO 模式

使用内部振荡器作为时钟源可以避免使用两个外部振荡器引脚，而将它们用作数字 I/O。目前有两种不同的配置：

- 在 INTIO1 模式， OSC2 引脚输出 Fosc/4 信号，而 OSC1 引脚则充当 RA7，用于数字输入和输出。
- 在 INTIO2 模式， OSC1 充当 RA7， OSC2 充当 RA6，两者都用于数字输入和输出。

2.6.2 INTOSC 输出频率

出厂时已校准了内部振荡器电路使之能够产生8 MHz的 INTOSC 输出频率。

INTRC 振荡器的工作独立于 INTOSC 时钟源。INTOSC 随电压和温度的变化并不会导致 INTRC 变化，反之亦然。

2.6.3 OSCTUNE 寄存器

内部振荡器的输出虽然已经过了工厂校准，但用户仍可在应用中对它进行调整。这是通过写 OSCTUNE 寄存器（寄存器2-1）完成的。在整个调节范围内灵敏度保持不变。

当修改了 OSCTUNE 寄存器后，INTOSC 频率将开始向新的频率变化。 INTRC 时钟将在 8 个时钟周期 （大约 8 * 32 μs = 256 μs）内达到新的频率。INTOSC 时钟会在 1 ms 内稳定下来。在此变化期间，代码会继续执行。不会有任何迹象表明时钟频率发生了改变。

OSCTUNE 寄存器也有 INTSRC 和 PLLEN 位，它们控制内部振荡器电路的某些功能。当选择了 31 kHz 频率后，用户可通过 INTSRC 位选择用作时钟源的内部振荡器。第 **2.7.1 节 “振荡器控制寄存器”**中对此进行了更详细地说明。

在内部振荡器模式下， PLLEN 位控制 PLL 倍频器的工作模式。

2.6.4 INTOSC 模式下的 PLL

内部振荡器电路可以通过使用 4x 倍频器来产生比一般内部振荡器所能产生的时钟速率更快的器件时钟。当使能时， PLL 最高可产生 32 MHz 的时钟。

与 HSPLL 模式不同，PLL 由软件控制。控制位 PLLEN（OSCTUNE<6>）用来使能或禁止其工作。

当器件被配置为使用内部振荡器电路作为其主时钟源时（FOSC3:FOSC0 = `1001` 或 `1000`），可以使用 PLL。此外，仅当选定的输出频率是 4 MHz 或 8 MHz（OSCCON<6:4> = `111` 或 `110`）时，PLL 才会工作。如果两个条件都不满足，则禁止 PLL。

PLLEN 控制位只有在使用 PLL 的内部振荡器模式下才有效。在所有其他模式下，它被强制为 `0` 并且无效。

2.6.5 INTOSC 频率漂移

出厂前内部振荡电路输出（INTOSC）被校准为 8 MHz。但是，此频率可能会随着 VDD 电压或温度的改变而发生漂移，这会影响控制器的运行。通过修改 OSCTUNE 寄存器的值可以调节 INTOSC 的频率。这对 INTRC 时钟源的频率没有影响。

调节 INTOSC 时钟源需要了解何时调节、调节的方向以及在某些情况下的调整量。在第 **2.6.5.1 节“用 EUSART 补偿”**和第 **2.6.5.2 节“用定时器补偿”**将讨论两种补偿技术，也可以使用其他技术。

寄存器 2-1： OSCTUNE：振荡器调节寄存器

R/W-0	R/W-0(1)	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN(1)	—	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **INTSRC**：内部振荡器低频源选择位
1 = 来自 8 MHz INTOSC 时钟源的31.25 kHz 器件时钟 （使能 256 分频）
0 = 直接来自 INTRC 内部振荡器的31 kHz 器件时钟

bit 6 **PLLEN**：INTOSC 的倍频器 PLL 使能位 (1)
1 = 为 INTOSC 使能 PLL （仅 4 MHz 和 8 MHz）
0 = 禁止 PLL

bit 5 未用：读为 0

bit 4-0 **TUN4:TUN0**：频率调节位
01111 = 最高频率
• •
• •
00001
00000 = 中心频率。振荡器模块运行在已校准后的频率上。
11111
• •
• •
10000 = 最低频率

注 1： 仅在某些振荡器配置中可用；其他情况下，此位不可用，并且读为 0。详细信息请参见**第 2.6.4 节 “INTOSC 模式下的 PLL”**。

2.6.5.1 用 EUSART 补偿

在异步模式下，当 EUSART 开始产生帧错误或者接收数据有错误时，可能需要对时钟频率进行调节。帧错误表示器件时钟的频率太高。要对此进行调节，可以减小 OSTUNE 寄存器中的值来降低时钟频率。另一方面，数据中有错误可能表明时钟速度太低。要进行补偿，可以增大 OSTUNE 寄存器中的值来提高时钟频率。

2.6.5.2 用定时器补偿

此技术是将器件时钟速度与某个参考时钟进行比较。可能要用到两个定时器；一个由外设时钟提供时钟源，而另一个由一个固定的参考源 （如 Timer1 振荡器）提供时钟源。

两个定时器都被清零，但由参考源提供时钟信号的定时器产生中断。当中断发生时，使用内部时钟源的定时器值被读取且两个定时器都被清零。如果使用内部时钟源的定时器的值大于期望值，则表示内部振荡器电路运行过快。要对此进行调整，需减小 OSCTUNE 寄存器中的值。

2.7 时钟源和振荡器切换

与以前的 PIC18 器件一样， PIC18F1230/1330 系列器件允许将器件时钟源从主振荡器切换到备用低频时钟源。 PIC18F1230/1330 器件提供了两个备用时钟源。当使能备用时钟源时，可以使用多种功耗管理工作模式。

基本上，这些器件有 3 种时钟源：

- 主振荡器
- 辅助振荡器
- 内部振荡器电路

主振荡器包括外部晶振和谐振器模式、外部 RC 模式、外部时钟模式和内部振荡器电路。特定的模式由 FOSC3:FOSC0 配置位定义。这些模式已在本章前面的内容中做过详细介绍。

辅助振荡器是指那些不与 OSC1 或 OSC2 引脚连接的外部时钟源。即使在控制器处于功耗管理模式时这些时钟源仍然可以继续工作。

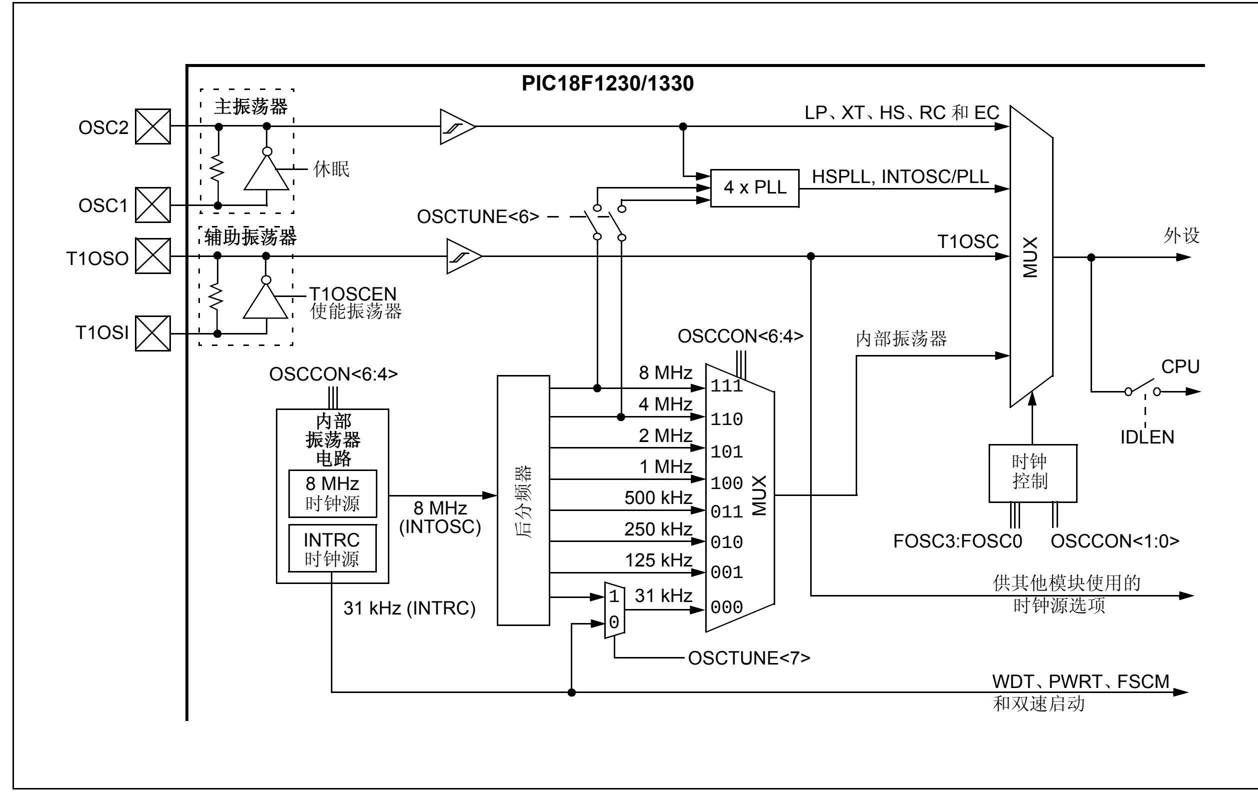
PIC18F1230/1330 器件将 Timer1 振荡器作为辅助振荡器。此振荡器在所有功耗管理模式下通常被用作实时时钟等功能的时基。

大部分情况下，在 T1OSO/T1CKI 和 T1OSI 引脚之间接有一个 32.768 kHz 的时钟晶振。与 LP 模式振荡器电路类似，每个引脚均都通过负载电容接地。在**第 12.2 节 "Timer1 振荡器"**中对 Timer1 振荡器进行了更详细地讨论。

除了作为主时钟源外，**内部振荡器电路**还可以作为功耗管理模式的时钟源。INTRC 时钟源也可以作为几种特殊功能部件的时钟源，例如 WDT 和故障保护时钟监视器。

图 2-8 显示了 PIC18F1230/1330 器件的时钟源。如需了解配置寄存器的详细信息，请参见**第 19.0 节 "CPU 的特殊功能"**。

图 2-8： PIC18F1230/1330 框图



 超前信息

2.7.1 振荡器控制寄存器

OSCCON 寄存器 （寄存器 2-2）控制全功耗模式和功耗管理模式下器件时钟的工作。

系统时钟选择位 SCS1:SCS0 选择时钟源。可用的时钟源有主时钟（由 FOSC3:FOSC0 配置位定义）、辅助时钟 （Timer1 振荡器）和内部振荡器电路。当写入一个或多个位后，有一段短的时钟转换间隔，然后，时钟源会立即改变。所有形式的复位均会使 SCS 位清零。

内部振荡器频率选择位 IRCF2:IRCF0 用于选择驱动器件时钟的内部振荡器电路的输出频率。这些频率可以是 INTRC时钟源的频率、INTOSC时钟源的频率（8 MHz）或 INTOSC 后分频器产生的几个频率之一 （31.25 kHz 至 4 MHz）。如果器件时钟源由内部振荡器电路提供，改变这些位的状态会使内部振荡器输出立即发生改变。当器件复位时，内部振荡器电路的默认输出频率设为 1 MHz。

当选择 31 kHz 的标称输出频率（IRCF2:IRCF0 = 000）时，用户可以选择充当时钟源的内部振荡器。这通过设置OSCTUNE寄存器中的INTSRC位（OSCTUNE<7>）来实现。将该位置 1 选择 INTOSC 作为时钟源。并通过使能 INTOSC 后分频器的 256 分频输出使输出频率为 31.25 kHz。将该位清零选择INTRC（标称值为31 kHz）作为时钟源。

该功能使用户能选择可调节且更精确的INTOSC作为时钟源，同时以非常低的时钟速率运行以节省功耗。无论 INTSRC 的设置如何，INTRC 总是作为看门狗定时器和故障保护时钟监视器之类部件的时钟源。

OSTS、IOFS 和 T1RUN 位指明当前提供器件时钟的是哪一个时钟源。OSTS 位置 1 表明振荡器起振定时器已超时且主时钟在主时钟模式下作为器件时钟源。 IOFS 位置 1 表明内部振荡器电路已稳定并在 RC 时钟模式下提供器件时钟。 T1RUN 位 （T1CON<6>）置 1 表明 Timer1 振荡器正在辅助时钟模式下提供器件时钟。在功耗管理模式下，任何时候这 3 个位只有一个会置 1。如果这些位都没有置 1，则表示当前时钟源是 INTRC，或内部振荡器电路刚刚起振且尚未稳定。

IDLEN 位确定当执行 SLEEP 指令时，器件是进入休眠模式还是某种空闲模式。

第 3.0 节“功耗管理模式”中将更详细地讨论 OSCCON 寄存器中的标志位和控制位的使用。

注	
1：	要选择辅助时钟源，必须使能 Timer1 振荡器。通过将 Timer1 控制寄存器中的 T1OSCEN 位 （T1CON<3>）置 1，可以使能 Timer1 振荡器。如果未使能 Timer1 振荡器，选择辅助时钟源的任何尝试都会被忽略。
2：	建议在Timer1振荡器稳定工作之后再选择辅助时钟源，否则当 Timer1 振荡器起振时可能会发生很长的延迟。

2.7.2 振荡器转换

PIC18F1230/1330 器件包含了防止在切换时钟源时发生时钟“毛刺”的电路。在时钟切换时，系统时钟会有短暂停顿。停顿的长度是旧时钟源的两个周期加上新时钟源的三到四个周期的和。此公式假设新时钟源是稳定的。

第 3.1.2 节 “进入功耗管理模式”中将更详细地讨论时钟转换。

寄存器 2-2： OSCCON：振荡器控制寄存器

R/W-0	R/W-1	R/W-0	R/W-0	R[(1)]	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **IDLEN**：空闲使能位

1 = 执行 SLEEP 指令后器件进入空闲模式
0 = 执行 SLEEP 指令后器件进入休眠模式

bit 6-4 **IRCF2:IRCF0**：内部振荡频率选择位

111 = 8 MHz （INTOSC 直接驱动时钟）
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz[(3)]
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz （来自 INTOSC 的 256 分频信号或直接来自 INTRC）[(2)]

bit 3 **OSTS**：振荡器起振延时状态位 [(1)]

1 = 振荡器起振定时器延时已经结束；主振荡器正在运行
0 = 振荡器起振定时器正在进行延时；主振荡器尚未准备就绪

bit 2 **IOFS** INTOSC 频率稳定位

1 = INTOSC 频率稳定
0 = INTOSC 频率不稳定

bit 1-0 **SCS1:SCS0**：系统时钟选择位

1x = 内部振荡器电路
01 = 辅助 （Timer1）振荡器
00 = 主振荡器

注 **1**： 复位状态取决于 IESO 配置位的状态。
2： 时钟源由 INTSRC 位 （OSCTUNE<7>）选择，请参见正文。
3： 复位时 INTOSC 的默认输出频率。

2.8 功耗管理模式对各种时钟源的影响

当选择了 PRI_IDLE 模式时，指定的主振荡器会继续运行而不会有任何停顿。对于所有其他的功耗管理模式，使用 OSC1 引脚的振荡器会被禁止。OSC1 引脚（以及由振荡器使用的 OSC2 引脚）将会停止振荡。

在辅助时钟模式（SEC_RUN和SEC_IDLE）下，Timer1 振荡器作为系统时钟源工作。如果需要，Timer1 振荡器也可以运行在所有功耗管理模式下为 Timer1 或 Timer3 提供时钟源。

在内部振荡器模式（RC_RUN 和 RC_IDLE）下，由内部振荡器电路提供器件时钟源。可以直接使用31 kHz的 INTRC 输出提供时钟或者使能它来支持多种特殊功能部件，这与功耗管理模式无关（欲知更多有关 WDT、故障保护时钟监视器和双速启动的信息，请参见**第 19.2 节“看门狗定时器（WDT）”**、**第 19.3 节“双速启动”**和**第 19.4 节“故障保护时钟监视器”**）。8 MHz 的 INTOSC 输出可以直接为器件提供时钟，或者先由后分频器进行分频再用作器件时钟。如果直接由 INTRC 输出提供时钟，则会禁止 INTOSC 输出。

如果选择了休眠模式，所有的时钟源都会停止。因为休眠模式关断了所有晶体管的开启电流，休眠模式能实现最小的器件电流消耗（仅泄漏电流）。

在休眠期间使能任何片上功能都将会增加休眠时的电流消耗。需要使能 INTRC 来支持 WDT 工作。 Timer1 振荡器可以用来为实时时钟提供时钟源。不需要器件时钟源的其他功能部件也可以工作（即 INTn 引脚和其他部件）。在 **第 22.0 节 “电气规范”**中列出了可能显著增加电流消耗的外设。

2.9 上电延时

有两个定时器控制上电延时，这样大部分应用都无需外接复位电路。上电延时可以确保在器件电源稳定（常规环境下）和主时钟稳定工作之前器件保持复位状态。如需更多有关上电延时的信息，请参见**第 4.5 节 “器件复位定时器”**。

第一个定时器是上电延时定时器（Power-up Timer，PWRT），在上电时提供固定的延时（表 22-10 中的参数 33）。通过清零（ = 0）PWRTEN 配置位可使能该定时器。

第二个定时器是振荡器起振定时器（Oscillator Start-up Timer，OST），用于在晶振稳定前使单片机保持在复位状态（LP、XT 和 HS 模式）。OST 通过计数 1024 个振荡周期后允许振荡器为器件提供时钟。

当选定 HSPLL 振荡器模式后，器件将在 HS 模式下的 OST 延时后额外保持 2 ms 的复位状态，这样可使 PLL 锁定输入时钟频率。

在上电复位后，有一个 TCSD（表 22-10 中的参数 38）的延时，允许控制器为执行指令做好准备。此延时与其他延时并行发生。将 EC、RC 或 INTIO 模式之一用作主时钟源时，这可能是惟一的延时。

表 2-3：　休眠模式下 OSC1 和 OSC2 引脚状态

振荡器模式	OSC1 引脚	OSC2 引脚
RC 和 INTIO1	悬空，经外接电阻拉为高电平	处于逻辑低电平（时钟的 4 分频输出）
RCIO	悬空，经外接电阻拉为高电平	配置为 PORTA 的 bit 6
INTIO2	配置为 PORTA 的 bit 7	配置为 PORTA 的 bit 6
ECIO	悬空，由外部时钟拉高	配置为 PORTA 的 bit 6
EC	悬空，由外部时钟拉高	处于逻辑低电平（时钟的 4 分频输出）
LP、 XT 和 HS	反馈反相器被禁止，输出静态电平	反馈反相器被禁止，输出静态电平

注：　如需了解由于休眠和 $\overline{\text{MCLR}}$ 复位引起的延时的信息，请参见**第 4.0 节 “复位”**中的表 4-2。

注：

3.0 功耗管理模式

PIC18F1230/1330 器件总共提供七种工作模式，可以更有效地进行功耗管理。这些工作模式提供了多种选项，帮助在资源受限的应用（即电池供电的器件）中节省功耗。

一共有三类功耗管理模式：

- 运行模式
- 空闲模式
- 休眠模式

这些类别决定为器件的哪些部分提供时钟，有时还定义时钟速率。运行和空闲模式可以使用三种时钟源（主时钟源、辅助时钟源或内部振荡器电路）中的任意一种；而休眠模式则不使用时钟源。

功耗管理模式具有几种在以前的 PIC® 器件上提供的节约功耗的功能。其中之一就是 PIC18 器件提供的时钟切换功能，允许控制器使用 Timer1 振荡器代替主振荡器。节省功耗的功能还包括所有 PIC 器件都提供的休眠模式，器件的所有时钟均在此模式下停止工作。

3.1 选择功耗管理模式

选择功耗管理模式时，需要确定是否为 CPU 提供时钟以及选择何种时钟源。IDLEN 位（OSCCON<7>）控制是否为 CPU 提供时钟，而 SCS1:SCS0 位（OSCCON<1:0>）选择时钟源。表 3-1 总结了各个模式下的位设置、时钟源和受影响的模块。

3.1.1 时钟源

SCS1:SCS0 位允许在三个时钟源中选择用于功耗管理模式的时钟。它们是：

- 主时钟，由 FOSC3:FOSC0 配置位定义
- 辅助时钟 （Timer1 振荡器）
- 内部振荡器电路 （用于 RC 模式）

3.1.2 进入功耗管理模式

通过装载 OSCCON 寄存器可以开始从一种功耗管理模式切换到另一种功耗管理模式。 SCS1:SCS0 位选择时钟源并决定使用运行模式还是空闲模式。更改这些位将导致立即切换到新的时钟源 （假定新的时钟源正在运行）。切换也可能会遇到时钟转换延时。**第 3.1.3 节“时钟转换和状态指示位”**和其后的章节将会讨论这些问题。

执行SLEEP指令可以触发进入功耗管理空闲模式或休眠模式。最后实际进入哪个模式由 IDLEN 位的状态决定。

更改功耗管理模式并不总是需要设置所有的位，需要设置哪些配置位取决于当前的模式和将要切换到的模式。在执行SLEEP指令之前更改振荡器选择位或更改IDLEN位可完成多种模式之间的转换。如果已经正确地配置了 IDLEN 位，可能只需执行 SLEEP 指令就可实现模式切换。

表 3-1： 功耗管理模式

模式	OSCCON 位		模块时钟		可用时钟和振荡器
	IDLEN<7>(1)	SCS1:SCS0 <1:0>	CPU	外设	
休眠模式	0	N/A	停止	停止	无——所有时钟被禁止
PRI_RUN	N/A	00	由时钟源驱动	由时钟源驱动	主时钟——LP、XT、HS、HSPLL、RC、EC 和内部振荡器电路 (2)。 这是正常的全功耗执行模式。
SEC_RUN	N/A	01	由时钟源驱动	由时钟源驱动	辅助时钟——Timer1 振荡器
RC_RUN	N/A	1x	由时钟源驱动	由时钟源驱动	内部振荡器电路 (2)
PRI_IDLE	1	00	停止	由时钟源驱动	主时钟——LP、 XT、 HS、 HSPLL、 RC 和 EC
SEC_IDLE	1	01	停止	由时钟源驱动	辅助时钟——Timer1 振荡器
RC_IDLE	1	1x	停止	由时钟源驱动	内部振荡器电路 (2)

注 1： IDLEN 的值在执行 SLEEP 指令时有效。

2： 包括 INTOSC 和 INTOSC 后分频器以及 INTRC 时钟源。

3.1.3 时钟转换和状态指示位

在两个时钟源之间进行切换所需的时间长度是旧时钟源的 2 个周期与新时钟源的 3 到 4 个周期的和。此公式假设新时钟源是稳定的。

以下 3 位用于指示当前的时钟源及其状态。它们是：

- OSTS （OSCCON<3>）
- IOFS （OSCCON<2>）
- T1RUN （T1CON<6>）

一般来说，在一个给定的功耗管理模式中，这 3 位中只有 1 位会置 1。当 OSTS 位置 1 时，表明由主时钟提供器件时钟。当 IOFS 位置 1 时，表明由 INTOSC 提供稳定的 8 MHz 时钟给分频器，由分频器驱动器件时钟。当 T1RUN 位置 1 时，表明由 Timer1 振荡器提供时钟。如果这些位均不置 1，则表明要么由 INTRC 时钟源为器件提供时钟信号，要么 INTOSC 时钟源尚未稳定。

如果用FOSC3:FOSC0配置位将内部振荡器电路配置为主时钟源，则在 PRI_RUN 或 PRI_IDLE 模式中，OSTS 和 IOFS 位可能同时置 1。这表示主时钟 （INTOSC 输出）正在产生稳定的 8 MHz 输出。进入工作频率相同的另一个 RC 功耗管理模式将清零 OSTS 位。

> 注 **1：** 在仅修改 IRCF 位时应该特别小心。如果 VDD 电压小于 3V，可以选择比低 VDD 电压所能支持的时钟速率更高的速率。违反 VDD/FOSC 规范会导致器件运行不正常。
>
> **2：** 执行SLEEP指令不一定会使器件进入休眠模式。该指令充当触发条件，根据 IDLEN 位的设置，使控制器进入休眠模式或某种空闲模式。

3.1.4 多条 SLEEP 命令

执行SLEEP指令后进入的具体功耗管理模式由指令执行时 IDLEN 位的当前设置决定。如果执行另一条 SLEEP 指令，器件将在随后进入由该指令执行时的当前 IDLEN 位指定的功耗管理模式。如果 IDLEN 位已更改，器件将进入由新的设置指定的新的功耗管理模式。

3.2 运行模式

在运行模式中，内核和外设的时钟均处于正常工作状态。这些模式之间的差异在于时钟源的不同。

3.2.1 PRI_RUN 模式

PRI_RUN 模式是单片机在正常情况下的全功耗执行模式。除非使能了双速启动 （详情请见**第 19.3 节 “双速启动”**），该模式也是器件复位后的默认模式。在此模式下， OSTS 位置 1。如果内部振荡器电路为主时钟源，IOFS 位可能被置 1 （见**第 2.7.1 节 “振荡器控制寄存器”**）。

3.2.2 SEC_RUN 模式

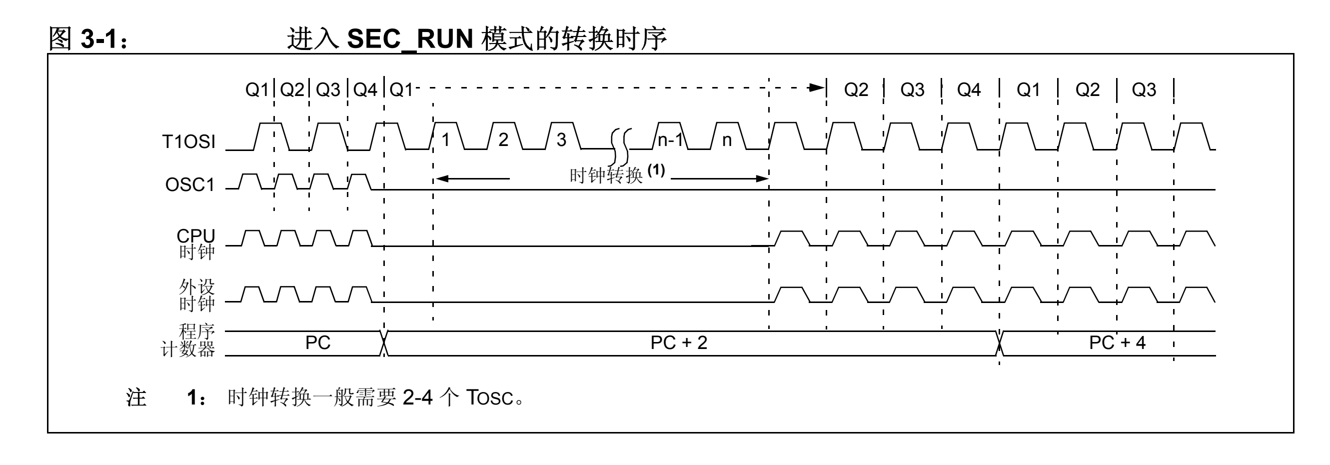
SEC_RUN 模式与其他 PIC18 器件提供的“时钟切换”功能兼容。在此模式下， CPU 和外设由 Timer1 振荡器提供时钟。这使得用户在仍使用高精度时钟源的情况下可获得较低的功耗。

通过将 SCS1:SCS0 位设置为 01 进入 SEC_RUN 模式。器件时钟源切换到 Timer1 振荡器（见图 3-1），关闭主振荡器， T1RUN 位（T1CON<6>）置 1 并且 OSTS 位清零。

> **注：** Timer1 振荡器应该在进入 SEC_RUN 模式之前已经开始运行。如果当SCS1:SCS0位被设置为 01 时， T1OSCEN 位未置 1，就不会进入 SEC_RUN 模式。如果使能了 Timer1 振荡器，但它尚未运行，器件时钟将会延时直到该振荡器起振。在这种情况下，最初的振荡器运行很不稳定，可能会导致无法预料的结果。

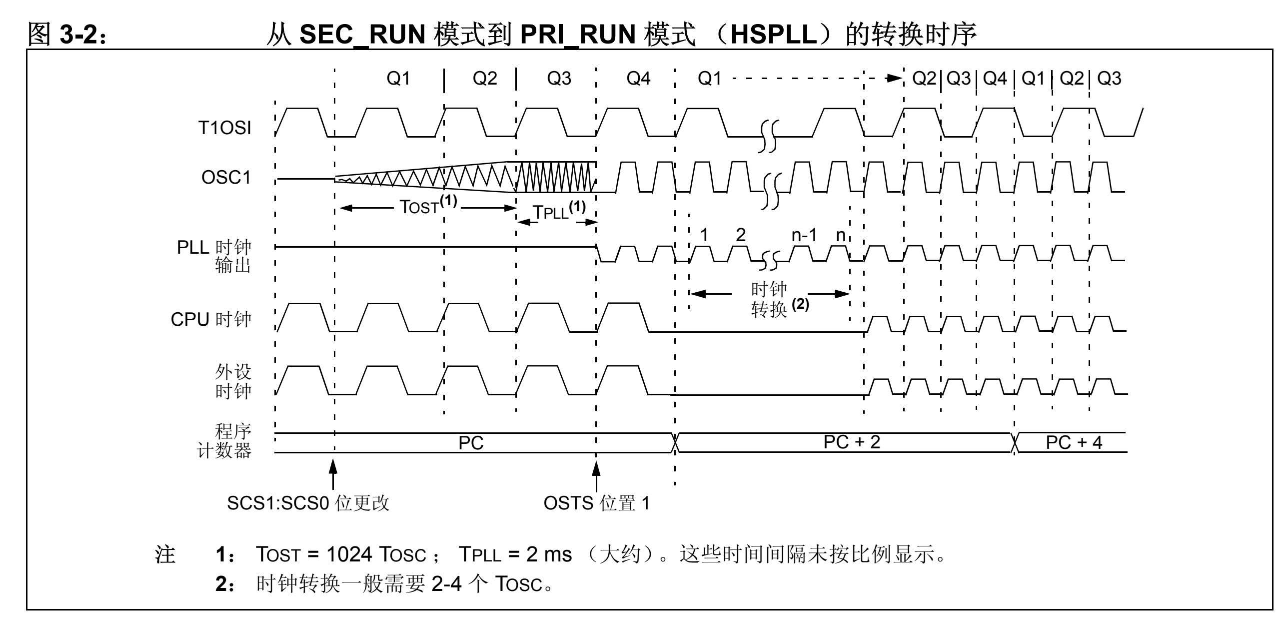
从 SEC_RUN 模式转换到 PRI_RUN 模式时，在主时钟起振期间外设和 CPU 继续将 Timer1 振荡器用作时钟源。当主时钟准备就绪以后，时钟切换回主时钟（见图 3-2）。当时钟切换完成后， T1RUN 位被清零， OSTS 位被置 1 并且由主时钟提供器件时钟。唤醒不会影响 IDLEN 和 SCS 位。 Timer1 振荡器继续运行。

图 3-1： 进入 SEC_RUN 模式的转换时序



注 1： 时钟转换一般需要 2-4 个 TOSC。

图 3-2： 从 SEC_RUN 模式到 PRI_RUN 模式 （HSPLL）的转换时序



注 1： TOST = 1024 TOSC ； TPLL = 2 ms （大约）。这些时间间隔未按比例显示。
2： 时钟转换一般需要 2-4 个 TOSC。

3.2.3 RC_RUN 模式

RC_RUN 模式中，内部振荡器电路使用 INTOSC 复用器作为 CPU 和外设的时钟源，此时主时钟关闭。在使用 INTRC 时钟源时，此模式是所有运行模式中最节省功耗的运行模式，同时仍然在执行代码。它非常适用于对时序精度要求不高或者不是一直需要高速时钟的应用。

如果主时钟源为内部振荡器电路 （INTRC 或 INTOSC），在执行期间，PRI_RUN 和 RC_RUN 这两种模式区别不大。但是在进入和退出 RC_RUN 模式时会发生时钟切换延时。因此，如果主时钟源为内部振荡器电路，建议不要使用 RC_RUN 模式。

通过将 SCS1 位置 1 可以进入此模式。虽然 SCS0 位的取值对操作没有影响，但还是建议将其清零，从而保证软件的兼容性。当时钟源切换到 INTOSC 复用器 （见图 3-3），主振荡器将被关闭并且 OSTS 位被清零。在任何时候更改 IRCF 位可以立即更改时钟速度。

注：	在仅修改 IRCF 位时应该特别小心。如果 VDD 电压小于 3V，可以选择比低 VDD 电压所能支持的时钟速率更高的速率。违反 VDD/FOSC 规范会导致器件运行不正常。

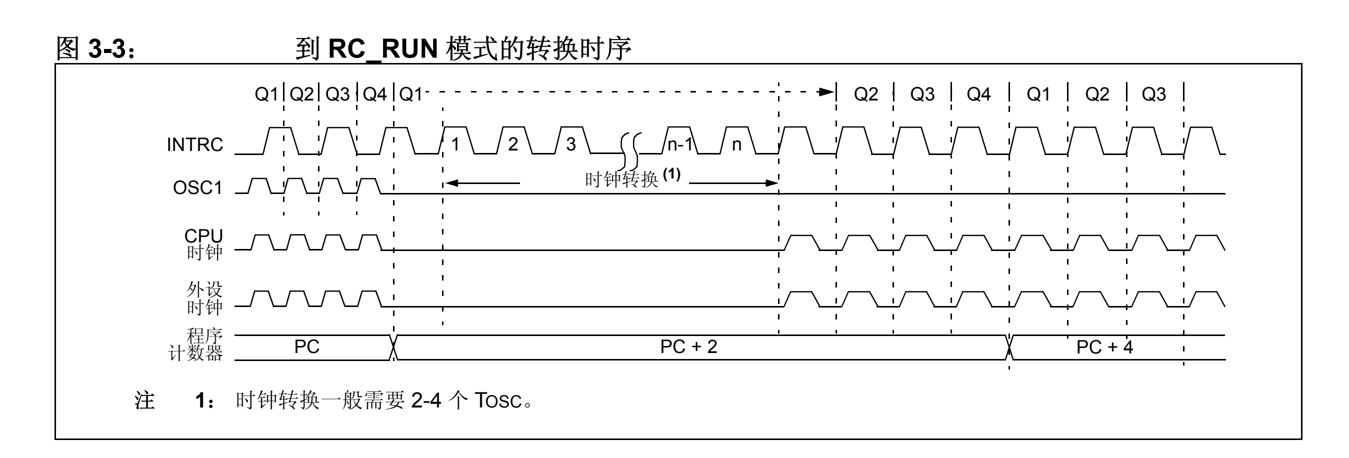
如果 IRCF 位和 INTSRC 位全部被清零，就会禁止 INTOSC 输出且 IOFS 位保持清零。此时不会有关于当前时钟源的任何指示。由 INTRC 时钟源提供器件时钟。

如果 IRCF 位从全零状态改变（因而使能 INTOSC 输出），或者 INTSRC 被置 1，在 INTOSC 输出变为稳定后 IOFS 位将被置 1。在一个 TIOBST 间隔之后，INTOSC 时钟源趋于稳定，此时器件时钟继续运行。

如果 IRCF 位在先前已被设置为一个非零值，或者在设置 SCS1 之前 INTSRC 已经置 1 并且 INTOSC 时钟源已经稳定，那么 IOFS 位将保持置 1 状态。

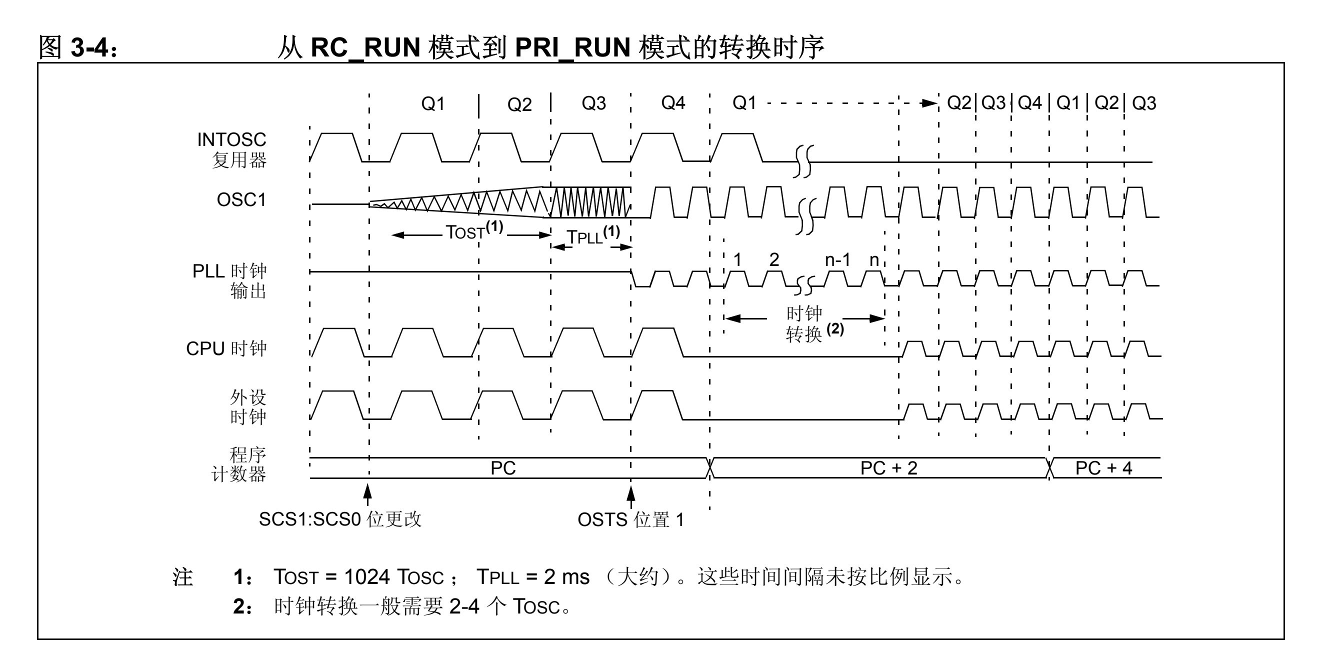
从 RC_RUN 模式转换到 PRI_RUN 模式时，在主时钟起振期间器件将继续使用 INTOSC 复用器作为时钟源。当主时钟准备好以后，时钟切换回主时钟（见图 3-4）。当时钟切换完成后，IOFS 位被清零，OSTS 位被置 1 并且由主时钟提供器件时钟。这种切换不会影响 IDLEN 和 SCS 位。如果使能了 WDT 或故障保护时钟监视器，INTRC 源将继续运行。

图 3-3： 到 RC_RUN 模式的转换时序



注 1： 时钟转换一般需要 2-4 个 TOSC。

图 3-4： 从 RC_RUN 模式到 PRI_RUN 模式的转换时序



注 1： TOST = 1024 TOSC ； TPLL = 2 ms （大约）。这些时间间隔未按比例显示。
2： 时钟转换一般需要 2-4 个 TOSC。

 超前信息

3.3 休眠模式

PIC18F1230/1330 器件的功耗管理休眠模式和所有其他 PIC 器件提供的传统休眠模式相同。通过清零 IDLEN 位（器件复位时的默认状态）并执行 SLEEP 指令即可进入此模式。这将关闭选定的振荡器（图 3-5）并将所有的时钟源状态位清零。

从其他模式进入休眠模式不需要时钟切换。这是因为单片机一旦进入休眠模式就不需要时钟了。如果选择了 WDT， INTRC 时钟源将继续运行。如果使能了 Timer1 振荡器， Timer1 也将继续运行。

在休眠模式中发生唤醒事件时（由于中断、复位或 WDT 超时），在 SCS1:SCS0 位选定的时钟源准备就绪之前，器件将没有时钟源（见图 3-6），但是如果使能了双速启动或故障保护时钟监视器，它将使用内部振荡器电路作为时钟源（见第 **19.0** 节 **"CPU 的特殊功能"**）。在这两种情况下，当主时钟提供器件时钟时， OSTS 位被置 1。唤醒不会影响 IDLEN 和 SCS 位。

3.4 空闲模式

空闲模式允许在外设继续工作时有选择地关闭单片机的 CPU。选择某种特定的空闲模式使用户能进一步管理功耗。

如果在执行 SLEEP 指令时，IDLEN 位被置为 1，外设将使用由 SCS1:SCS0 位选择的时钟作为时钟源，而 CPU 将没有时钟源。时钟源状态位不受影响。将 IDLEN 置 1 并执行 SLEEP 指令将从给定的运行模式快速切换到相应的空闲模式。

如果选择了 WDT， INTRC 时钟源将继续运行。如果使能了 Timer1 振荡器， Timer1 也将继续运行。

由于 CPU 没有执行指令，器件只能通过中断、WDT 超时或复位从任一空闲模式退出。当发生唤醒事件时，会经过一段长度为 TCSD（表 22-10 中的参数 38）的延时，供 CPU 为执行代码做准备，并在此段延时结束后执行代码。当 CPU 开始执行代码时，它将沿用与当前空闲模式相同的时钟源。例如，当从 RC_IDLE 模式唤醒时，将使用内部振荡器电路作为 CPU 和外设的时钟源（即 RC_RUN 模式）。唤醒不会影响 IDLEN 和 SCS 位。

当处于任何空闲模式或休眠模式时， WDT 超时将导致 WDT 唤醒到由 SCS1:SCS0 位的当前值指定的运行模式。

图 3-5： 进入休眠模式的转换时序

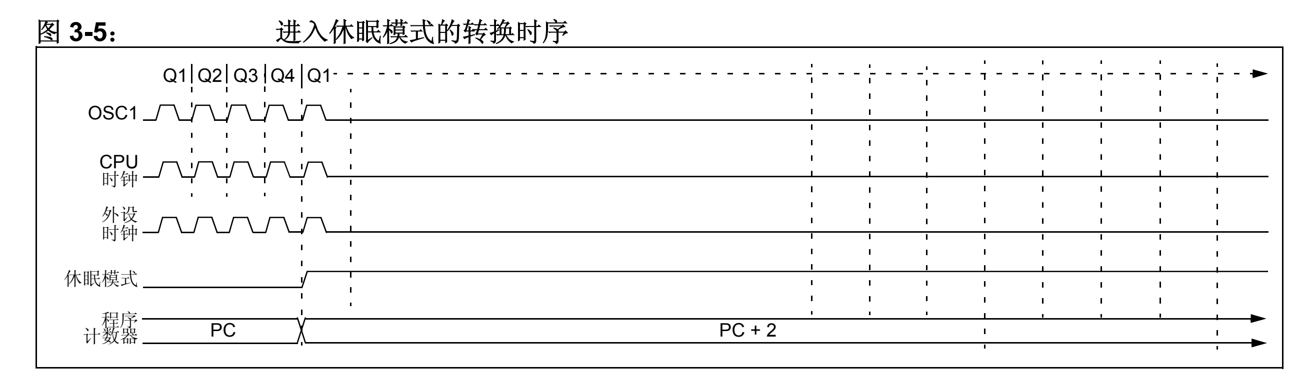
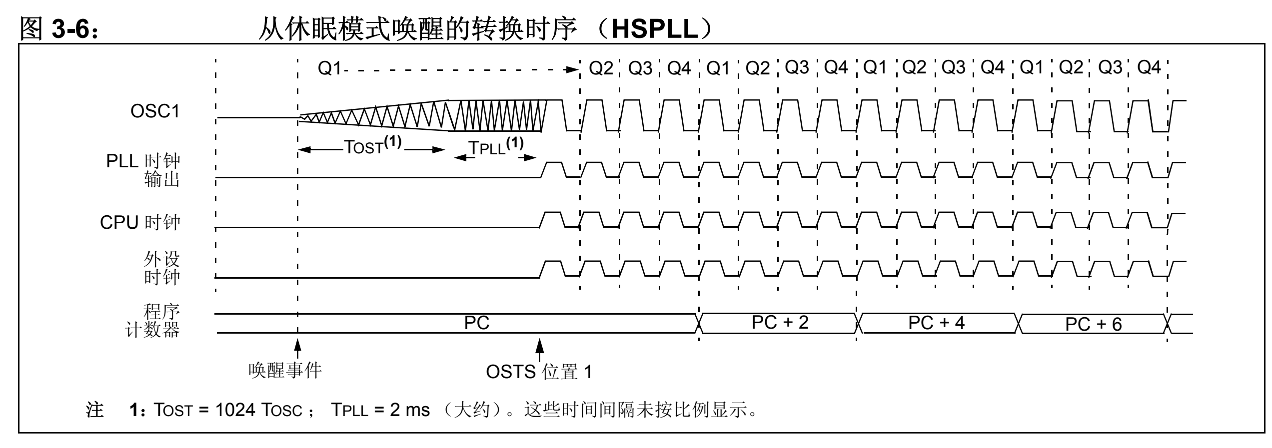


图 3-6： 从休眠模式唤醒的转换时序 （**HSPLL**）



注 1：T_{OST} = 1024 T_{OSC}； T_{PLL} = 2 ms（大约）。这些时间间隔未按比例显示。

3.4.1 PRI_IDLE 模式

此模式是在三种低功耗空闲模式中唯一不会禁止主器件时钟的模式。对于时间敏感的应用来说，由于时钟源不需要“热身”或从其他振荡器转换，选用此模式可以有更加精确的主时钟源，并以最快的速度恢复器件运行。

通过将IDLEN位置1并执行SLEEP指令可以从PRI_RUN模式进入PRI_IDLE模式。如果器件处于其他运行模式，请先将IDLEN位置1，然后将SCS位清零并执行SLEEP指令。虽然禁止了CPU，外设仍继续由FOSC3:FOSC0配置位指定的主时钟源提供时钟。OSTS位保持置1（见图 3-7）。

当发生唤醒事件时，由主时钟源为 CPU 提供时钟。在唤醒事件和代码执行开始之间需要一个 T_{CSD} 的延时，以使 CPU 做好执行指令的准备。在唤醒之后，OSTS 位保持置 1。唤醒不会影响 IDLEN 和 SCS 位。（见图 3-8）。

3.4.2 SEC_IDLE 模式

在 SEC_IDLE 模式中， CPU 被禁止，但外设继续将 Timer1振荡器作为时钟源。将IDLEN置1并执行SLEEP指令从SEC_RUN模式进入SEC_IDLE模式。如果器件处于另一运行模式，先将 IDLEN 置 1，然后将SCS1:SCS0 位置为 01，并执行 SLEEP 也能进入SEC_IDLE 模式。当时钟源切换到 Timer1 振荡器时，主振荡器关闭， OSTS 位清零且 T1RUN 位置 1。

当唤醒事件发生时，外设继续将 Timer 振荡器作为时钟源。唤醒事件发生后经过一个 T_{CSD} 的时间间隔， CPU 开始执行代码并使用 Timer1 振荡器作为其时钟源。唤醒不会影响 IDLEN 和 SCS 位。Timer1 振荡器继续运行（见图 3-8）。

> **注：** Timer1 振荡器应该在进入 SEC_IDLE 模式之前就已经运行了。如果执行 SLEEP 指令时T1OSCEN位没有置1，就会忽略SLEEP指令并且不会进入 SEC_IDLE 模式。如果使能了 Timer1 振荡器，但它尚未运行，外设时钟将会延时直到该振荡器起振。在这种情况下，最初的振荡器运行很不稳定，可能会导致无法预料的结果。

图 3-7： 进入空闲模式的转换时序

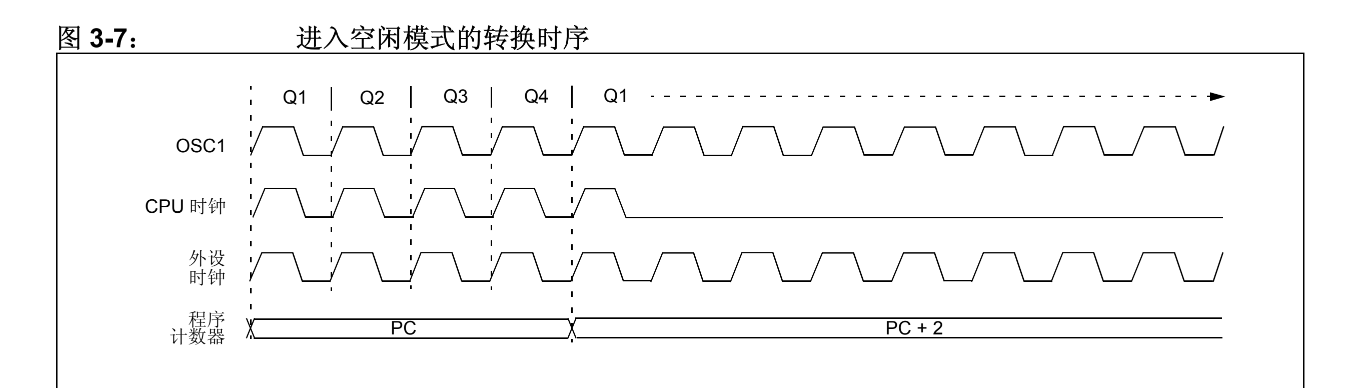
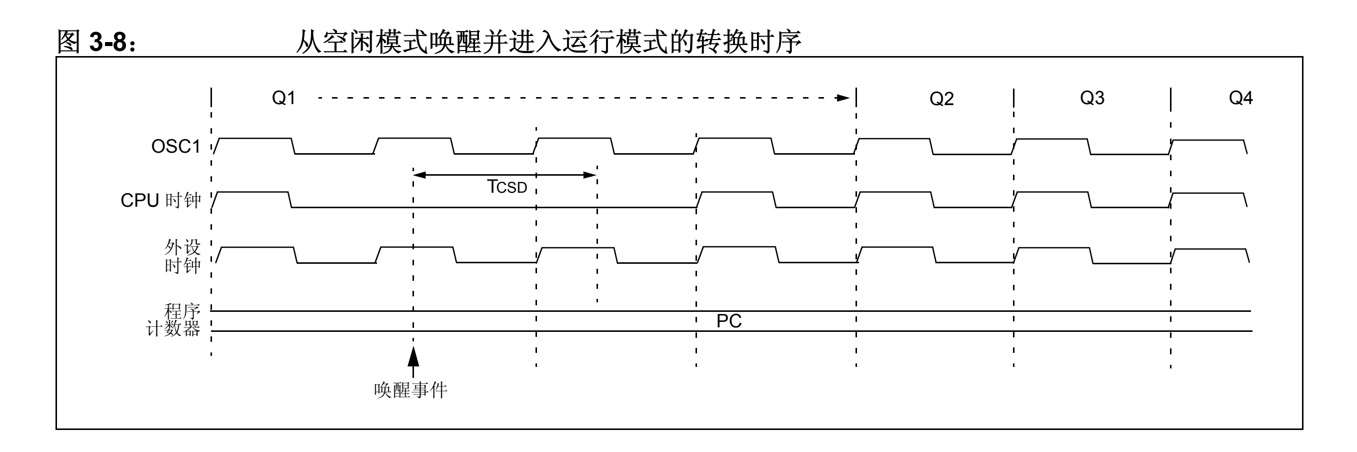


图 3-8： 从空闲模式唤醒并进入运行模式的转换时序



 超前信息

3.4.3 RC_IDLE 模式

在 RC_IDLE 模式下，CPU 被禁止，但外设仍继续使用内部振荡器电路的 INTOSC 复用器作为时钟源。该模式允许在空闲期间对功耗进行控制。

通过将IDLEN位置1并执行SLEEP指令可以从RC_RUN模式进入此模式。如果器件处于另一种运行模式，可以先将 IDLEN 位置 1，然后再将 SCS1 位置 1 并执行 SLEEP。虽然 SCS0 的值常常被忽略，但仍建议将其清零，这将保证与今后器件的软件兼容性。通过在执行 SLEEP 指令之前修改 IRCF 位可以为 INTOSC 复用器选择更高的时钟频率。当时钟源切换到 INTOSC 复用器，主振荡器被关闭， OSTS 位被清零。

如果 IRCF 位被设置为任何非零值，或者 INTSRC 位被置 1，就会使能 INTOSC 输出。在一个 T_{IOBST} 间隔（表 22-10 中的参数 39）之后， INTOSC 输出趋于稳定，然后 IOFS 位置 1。外设的时钟继续运行直到 INTOSC 时钟源趋于稳定。如果 IRCF 位在先前已被设置为一个非零值，或者 INTSRC 在执行 SLEEP 之前就已经置 1 并且 INTOSC 源已经稳定，那么 IOFS 位将保持置 1。如果 IRCF 和 INTSRC 位全部清零，就不会使能 INTOSC 输出， IOFS 位将保持清零状态，此时将不会有当前时钟源的任何指示。

当唤醒事件发生时，外设继续将 INTOSC 复用器作为时钟源。唤醒事件发生后经过一个T_{CSD}的时间间隔，CPU开始执行代码并使用 INTOSC 复用器作为其时钟源。唤醒不会影响 IDLEN 和 SCS 位。如果使能了 WDT 或故障保护时钟监视器， INTRC 源将继续运行。

3.5 从空闲和休眠模式退出

中断、复位或 WDT 超时均可用作从休眠模式或任何空闲模式退出的触发条件。本节将讨论从功耗管理模式退出的触发方式。在每种功耗管理模式中我们还讨论了时钟源子系统的作用（见**第 3.2 节 "运行模式"**、**第 3.3 节 "休眠模式"**和**第 3.4 节 "空闲模式"**）。

3.5.1 通过中断退出

任何可用的中断源都可导致器件从空闲模式或休眠模式退出到运行模式。要使能此功能，必须将 INTCON 或 PIE 寄存器中的中断允许位置 1。当相应的中断标志位置 1 时，启动退出序列。

当通过中断从空闲或休眠模式退出时，如果GIE/GIEH位（INTCON<7>）置 1，代码就会跳转到中断向量处执行。否则，代码就会顺序执行或恢复而不发生跳转（见**第 10.0 节 "中断"**）。

唤醒事件之后需要一个固定的 T_{CSD} 间隔的延时，器件才会退出休眠和空闲模式。CPU 需要此延时来为执行代码做准备。在此延时后的第一个时钟周期恢复指令执行。

3.5.2 通过 WDT 超时退出

WDT 根据超时发生时器件所处的不同功耗管理模式会进行不同的操作。

如果器件不在执行代码（所有空闲模式和休眠模式），超时将导致从功耗管理模式退出（见**第 3.2 节 "运行模式"**和**第 3.3 节 "休眠模式"**）。如果器件正在执行代码（所有运行模式），超时将导致 WDT 复位（见**第 19.2 节 "看门狗定时器（WDT）"**）。

执行 SLEEP 或 CLRWDT 指令，或者当前选择的时钟源失效（如果使能了故障保护时钟监视器）以及修改 OSCCON 寄存器中的 IRCF 位（如果器件时钟源为内部振荡器电路），均将清零 WDT 定时器和后分频器。

3.5.3 通过复位退出

通常，器件通过振荡器起振定时器（OST）保持在复位状态，直到主时钟就绪。主时钟就绪后，OSTS 位置 1，器件开始执行代码。如果以内部振荡器电路作为新的时钟源，则 IOFS 位将置 1。

从复位状态退出到开始执行代码期间的延迟时间由唤醒前后的时钟源以及主时钟振荡器的类型（如果新的时钟源为主时钟）决定。表 3-2 为退出延时的汇总。

可以在主时钟就绪之前开始执行代码。如果使能了双速启动（见**第 19.3 节 "双速启动"**）或故障保护时钟监视器（见**第 19.4 节 "故障保护时钟监视器"**），器件可以在复位源被清除之后马上开始执行代码。由内部振荡器电路驱动的 INTOSC 复用器作为代码执行的时钟源。执行代码时，由内部振荡器电路提供时钟源直到主时钟就绪，或者在主时钟就绪前进入功耗管理模式，主时钟会在随后关闭。

3.5.4 无需振荡器起振延时的退出

从某些功耗管理模式退出不会引起 OST 延迟。有两种情形：

- 主时钟源一直工作的 PRI_IDLE 模式
- 主时钟源不是 LP、 XT、 HS 或 HSPLL 中的任意一种模式。

在这些情况下，主时钟源不需要振荡器起振延时，因为它已经在运行（PRI_IDLE），或者它本来就不需要振荡器起振延时（RC、EC 和 INTIO 振荡模式）。然而，当器件退出休眠和空闲模式时，在唤醒事件后仍需要一个固定的延时 TCSD，以便让 CPU 为执行代码做好准备。在此延时后的第一个时钟周期恢复指令执行。

表 3-2： 通过复位从休眠模式或任何空闲模式唤醒时的退出延时 （按时钟源分类）

唤醒前的时钟源	唤醒后的时钟源	退出延时	时钟就绪状态位（**OSCCON**）
主器件时钟（PRI_IDLE 模式）	LP、 XT 和 HS	TCSD[1]	OSTS
	HSPLL		
	EC 和 RC		
	INTOSC[2]		IOFS
T1OSC	LP、 XT 和 HS	TOST[3]	OSTS
	HSPLL	TOST + t_{rc}[3]	
	EC 和 RC	TCSD[1]	
	INTOSC[1]	TIOBST[4]	IOFS
INTOSC[3]	LP、 XT 和 HS	TOST[4]	OSTS
	HSPLL	TOST + t_{rc}[3]	
	EC 和 RC	TCSD[1]	
	INTOSC[1]	无	IOFS
无（休眠模式）	LP、 XT 和 HS	TOST[3]	OSTS
	HSPLL	TOST + t_{rc}[3]	
	EC 和 RC	TCSD[1]	
	INTOSC[1]	TIOBST[4]	IOFS

注 **1：** 当从休眠模式和空闲模式唤醒时都需要 TCSD （参数 38）延时，同时伴有所需的其他延时 （见第 **3.4** 节 **“空闲模式”**）。复位时， INTOSC 默认为 1 MHz。

2： 包括 INTOSC 8 MHz 时钟源和后分频器产生的频率。

3： TOST 是振荡器起振定时器的延迟时间 （参数 32）。 t_{rc} 是 PLL 锁定延时定时器的延迟时间 （参数 F12），也称 TPLL。

4： 在 INTOSC 稳定周期 TIOBST （参数 39）内，代码继续执行。

4.0 复位

PIC18F1230/1330 器件有以下几种不同的复位方式：

a) 上电复位 （POR）
b) 正常工作状态下的 $\overline{\text{MCLR}}$ 复位
c) 功耗管理模式下的 $\overline{\text{MCLR}}$ 复位
d) 看门狗定时器 （WDT）复位 （执行程序期间）
e) 可编程欠压复位 （BOR）
f) RESET 指令
g) 堆栈满复位
h) 堆栈下溢复位

本节将讨论由 $\overline{\text{MCLR}}$、POR 和 BOR 产生的各种复位以及各种起振定时器的操作。堆栈复位事件将在**第 5.1.2.4 节"堆栈满和下溢复位"**中讨论。WDT 复位将在**第 19.2 节 "看门狗定时器 （WDT）"**中讨论。

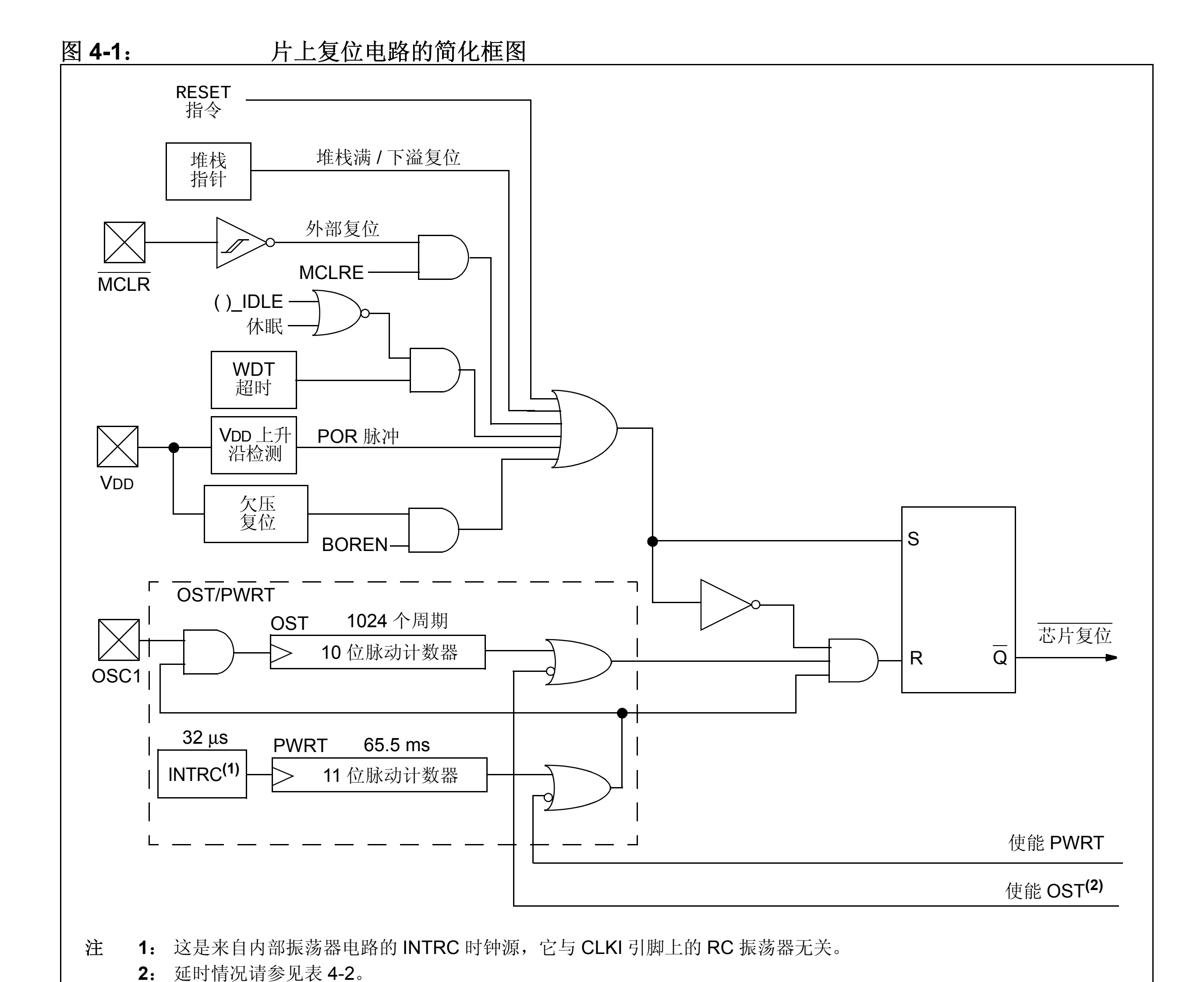
图 4-1 给出了片上复位电路的简化框图。

4.1 RCON 寄存器

可通过 RCON 寄存器 （寄存器 4-1）跟踪器件复位事件。该寄存器的低 5 位表示特定的复位事件是否已经发生。在大部分情况下，只能通过事件将这些位清零，而且必须在随后的应用程序中将它们置 1。读这 5 位标志位可以知道刚刚发生过的复位的类型。**第 4.6 节 "寄存器的复位状态"**中对此进行了更详细地说明。

RCON 寄存器中还有用于设置中断优先级的控制位（IPEN）和用于对 BOR 进行软件控制的控制位（SBOREN）。**第 10.0 节 "中断"**将讨论中断优先级。**第 4.4 节 "欠压复位 （BOR）"**中将介绍 BOR。

图 4-1： 片上复位电路的简化框图



注 1： 这是来自内部振荡器电路的 INTRC 时钟源，它与 CLKI 引脚上的 RC 振荡器无关。
2： 延时情况请参见表 4-2。

寄存器 4-1： RCON：复位控制寄存器

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	RI	TO	PD	POR	BOR
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **IPEN**：中断优先级使能位
1 = 使能中断优先级
0 = 禁止中断优先级 （PIC16CXXX 兼容模式）

bit 6 **SBOREN**：BOR 软件使能位 (1)
如果 BOREN1:BOREN0 = 01：
1 = 使能 BOR
0 = 禁止 BOR
如果 BOREN1:BOREN0 = 00、 10 或 11：
禁止该位，其读为 0。

bit 5 未用：读为 0

bit 4 **RI**：RESET 指令标志位
1 = 未执行 RESET 指令 （仅由固件置 1）
0 = 已执行 RESET 指令，导致器件复位 （必须在发生复位后用软件置 1）

bit 3 **TO**：看门狗超时标志位
1 = 通过上电、CLRWDT 指令或 SLEEP 指令置 1
0 = 发生了 WDT 超时

bit 2 **PD**：掉电检测标志位
1 = 通过上电或 CLRWDT 指令置 1
0 = 通过执行 SLEEP 指令置 1

bit 1 **POR**：上电复位状态位 (2)
1 = 未发生上电复位 （仅由固件置 1）
0 = 发生了上电复位 （必须在发生上电复位后由软件置 1）

bit 0 **BOR**：欠压复位状态位
1 = 未发生欠压复位 （仅由固件置 1）
0 = 发生了欠压复位 （必须在欠压复位发生后由软件置 1）

注 1： 如果使能 SBOREN 位 ，其复位状态为 1，否则为 0。
2： POR 的实际复位值由器件复位的类型决定。欲知更多信息，请参见本寄存器说明后面的 “注”和**第 4.6 节 “寄存器的复位状态”**。

注 1：建议在检测到上电复位后，将 POR 位置 1，以便继续检测后续的上电复位。
2：当 BOR 为 0 且 POR 为 1 时（假定在 POR 之后立即由软件将 POR 位置 1），可以说已发生了欠压复位。

4.2 主清零 （MCLR）

MCLR 引脚提供了用外部硬件触发器件复位的方法。将该引脚拉低可以产生复位信号。这些器件在 MCLR 复位路径上有一个噪声滤波器，该滤波器可以检测并滤除小的干扰脉冲。

任何内部复位，包括 WDT 复位，都不能将 MCLR 引脚驱动为低电平。

在 PIC18F1230/1330 器件中，可以用 MCLRE 配置位禁止 MCLR 输入。当禁止 MCLR 时，该引脚成为一个数字输入引脚。如需更多信息，请参见**第 9.1 节 “PORTA、 TRISA 和 LATA 寄存器”**。

4.3 上电复位 （POR）

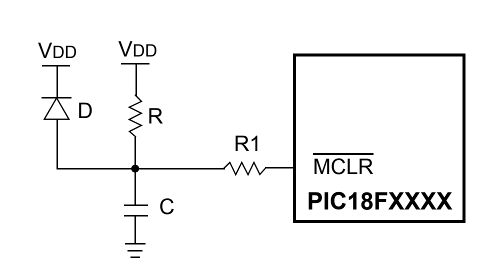
只要当 VDD 超过设定的门限值后，就会在片上产生上电复位脉冲。这使得 VDD 达到满足器件正常工作的数值时，器件会初始化并启动。

要使用 POR 电路，需要将 MCLR 引脚通过一个电阻（阻值为 1 kΩ 到 10 kΩ）连接到 VDD。这样可以省去产生上电复位延时通常需要的外部 RC 元件。VDD 的最小上升速率在参数 D004 中指定。对于上升速率缓慢的情况，请参见图 4-2。

当器件开始正常工作（即退出复位状态）时，必须满足特定的工作参数要求（电压、频率和温度等），才能确保其正常工作。如果不满足这些条件，那么器件必须保持在复位状态，直到满足工作条件为止。

POR 事件由 POR 位（RCON<1>）捕获。每当发生 POR 时，该位的状态就会被置为 0，任何其他的复位事件均不能改变它。POR 不能被硬件复位为 1。要捕获多个事件，用户必须在 POR 之后用软件将该位复位为 1。

图 4-2： 外部上电复位电路 （VDD 慢速上电）



注 1：仅当 VDD 上电速率过慢时才需要外部上电复位电路。当 VDD 掉电时，二极管 D 帮助电容迅速放电。

2：建议 R < 40 kΩ，确保电阻 R 两端的电压降符合器件的电气规范。

3：R1 ≥ 1 kΩ 将限制电流从外部电容 C 流入 MCLR，以避免由于静电放电（Electrostatic Discharge，ESD）或电超载（Electrical Overstress，EOS）导致 MCLR/VPP 引脚损坏。

4.4 欠压复位 （BOR）

PIC18F1230/1330 器件包含一个 BOR 电路，它将为用户提供一系列配置和节能选项。BOR由BORV1:BORV0 和 BOREN1:BOREN0 配置位控制。总共有四种 BOR 配置，归纳在表 4-1 中。

BOR 门限值由 BORV1:BORV0 位设置。如果使能 BOR （BOREN1:BOREN0 为非零值），当 V_{DD} 跌落到 V_{BOR} （参数 D005）以下的时间超过 T_{BOR} （参数 35）时就会复位器件。如果 V_{DD} 降到 V_{BOR} 以下的时间小于 T_{BOR}，可能不一定发生复位。芯片将保持欠压复位状态，直至 V_{DD} 电压上升到 V_{BOR} 以上。

如果使能上电延时定时器，则它将在 V_{DD} 上升到超过 V_{BOR} 之后开始工作，并使芯片在延时 T_{PWRT} （参数 33）期间保持复位。如果在上电延时定时器运行过程中，V_{DD} 电压降到 V_{BOR} 以下，芯片将重新回到欠压复位状态且将初始化上电延时定时器。一旦 V_{DD} 电压上升到 V_{BOR} 以上，上电延时定时器将重新执行延时操作。

BOR 和上电延时定时器 （PWRT）是独立配置的。使能 BOR 复位并不会自动使能 PWRT。

4.4.1 用软件使能 BOR

当 BOREN1:BOREN0 = 01 时，用户可以用软件使能或禁止 BOR。这可通过控制位 SBOREN （RCON<6>）完成。如前所述，将 SBOREN 置 1 可使能 BOR。清零 SBOREN 将完全禁止 BOR。 SBOREN 位只用于该模式，其他情况下它的读取值为 0。

用软件控制BOR位可使用户能更灵活地定制应用程序，而无需通过对器件重新编程来更改 BOR 配置。它还允许用户通过减少 BOR 消耗的电流，用软件调节器件的功耗。虽然 BOR 的电流通常很小，但是它可能对低功耗应用有一些影响。

> **注：** 即使当 BOR 受软件控制时， BOR 复位电平仍然将由 BORV1:BORV0 配置位设置。该值不能用软件更改。

4.4.2 检测 BOR

使能 BOR 后，当发生 BOR 或 POR 事件时，$\overline{\text{BOR}}$ 位都会复位为 0。因此仅通过读 $\overline{\text{BOR}}$ 的状态很难确定是否发生了 BOR 事件。更可靠的方法是同时检查 $\overline{\text{POR}}$ 和 $\overline{\text{BOR}}$ 的状态。假定在发生 POR 事件后，立即用软件将 $\overline{\text{POR}}$ 位置 1。如果 $\overline{\text{BOR}}$ 为 0 而 $\overline{\text{POR}}$ 为 1，就可以明确断定已经发生了 BOR 事件。

4.4.3 在休眠模式下禁止 BOR

当 BOREN1:BOREN0 = 10 时，BOR 保持受硬件控制状态并且像前面描述的那样工作。每当器件进入休眠模式时，就会自动禁止 BOR。当器件返回到任何其他工作模式时，又将自动重新使能 BOR。

此模式使应用程序能在有效地执行代码的同时从欠压状态恢复，这也是器件最需要 BOR 保护的状况。同时，通过消除增加的 BOR 电流，可以省去休眠模式下的额外功耗。

表 4-1： BOR 配置

BOR 配置		SBOREN 的状态 （RCON<6>）	BOR 操作
BOREN1	BOREN0		
0	0	不可用	禁止 BOR ；必须对配置位重新编程才能使能 BOR。
0	1	可用	用软件使能 BOR ；工作模式由 SBOREN 控制。
1	0	不可用	在运行和空闲模式下用硬件使能 BOR，在休眠模式下禁止。
1	1	不可用	用硬件使能 BOR ；必须对配置位重新编程才能禁止 BOR。

4.5 器件复位定时器

PIC18F1230/1330 器件包含了三个独立的片上定时器，有助于调节上电复位过程。它们的主要功能是确保代码执行之前系统时钟的稳定性。这些定时器是：

- 上电延时定时器 （PWRT）
- 振荡器起振定时器 （OST）
- PLL 锁定延时定时器

4.5.1 上电延时定时器 （PWRT）

PIC18F1230/1330 器件的上电延时定时器 （PWRT）是一个 11 位计数器，使用 INTRC 时钟源作为时钟输入。产生约 2048 x 32 µs = 65.6 ms 的延迟时间。当 PWRT 计数时，器件保持在复位状态。

上电延迟时间取决于 INTRC 时钟，并且由于温度和工艺的不同，不同芯片的延迟时间也各不相同。详情请参见 DC 参数 33。

通过清零配置位 $\overline{\text{PWRTEN}}$ 使能 PWRT。

4.5.2 振荡器起振定时器 （OST）

在 PWRT 延时 （参数 33）结束以后，由振荡器起振定时器 （OST）提供一个 1024 个振荡周期 （OSC1 输入）的延时，从而确保晶振或谐振器已经起振并稳定工作。

只有在 XT、LP、HS 和 HSPLL 模式下，且仅当发生上电复位或从大多数功耗管理模式退出时，才启动OST延时。

4.5.3 PLL 锁定延时

当在 PLL 模式下使能 PLL 时，上电复位后的延时时序与其他振荡器模式略有不同。使用一个独立的定时器来提供一段足够让 PLL 锁定主振荡器频率的固定延时。PLL 锁定延时 （TPLL）通常为 2 ms，且在振荡器起振延时后发生。

4.5.4 延时序列

上电延时序列如下：

1. POR脉冲清零后，启动PWRT延时（如果使能）。
2. 然后， OST 被激活。

总延时时间取决于振荡器的配置和PWRT的状态。图 4-3、图 4-4、图 4-5、图 4-6 和图 4-7 各自描述了不同的上电延时时序，其中上电延时定时器均被使能，并且器件工作在HS振荡器模式下。图 4-3 至图 4-6 也适用于在XT或LP模式下工作的器件。对于工作在 RC 模式下并禁止 PWRT 的器件，将没有延时。

由于延时是由 POR 脉冲触发的，因此如果 $\overline{\text{MCLR}}$ 保持足够长时间的低电平，所有延时都将结束。将 $\overline{\text{MCLR}}$ 电平拉高后程序将立即开始执行代码 （图 4-5）。这对于测试或同步多个并行工作的 PIC18FXXXX 器件来说非常有用。

表 4-2： 不同情况下的延时

振荡器配置	上电复位 (2) 和欠压复位		从功耗管理模式退出
	$\overline{\text{PWRTEN}}$ = 0	$\overline{\text{PWRTEN}}$ = 1	
HSPLL	66 ms(1) + 1024 Tosc + 2 ms(2)	1024 Tosc + 2 ms(2)	1024 Tosc + 2 ms(2)
HS、 XT 或 LP	66 ms(1) + 1024 Tosc	1024 Tosc	1024 Tosc
EC 或 ECIO	66 ms(1)	—	—
RC 或 RCIO	66 ms(1)	—	—
INTIO1 或 INTIO2	66 ms(1)	—	—

注 **1：** 66 ms （65.5 ms）是上电延时定时器 （PWRT）延迟时间的标称值。
2： 2 ms 是 PLL 锁定所需的标称时间。

图 4-3： 上电延时时序（$\overline{\text{MCLR}}$ 连接到 VDD，VDD 电压上升时间为 < TPWRT）

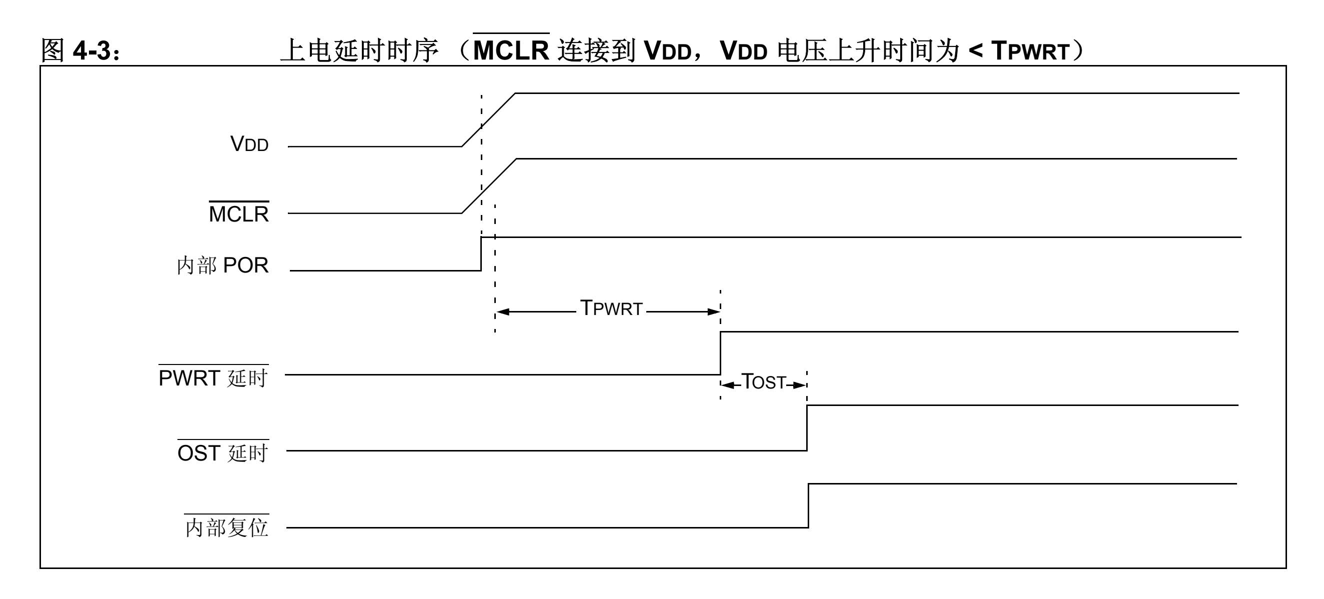


图 4-4： 上电延时时序（$\overline{\text{MCLR}}$ 未连接到 VDD）：情形 1

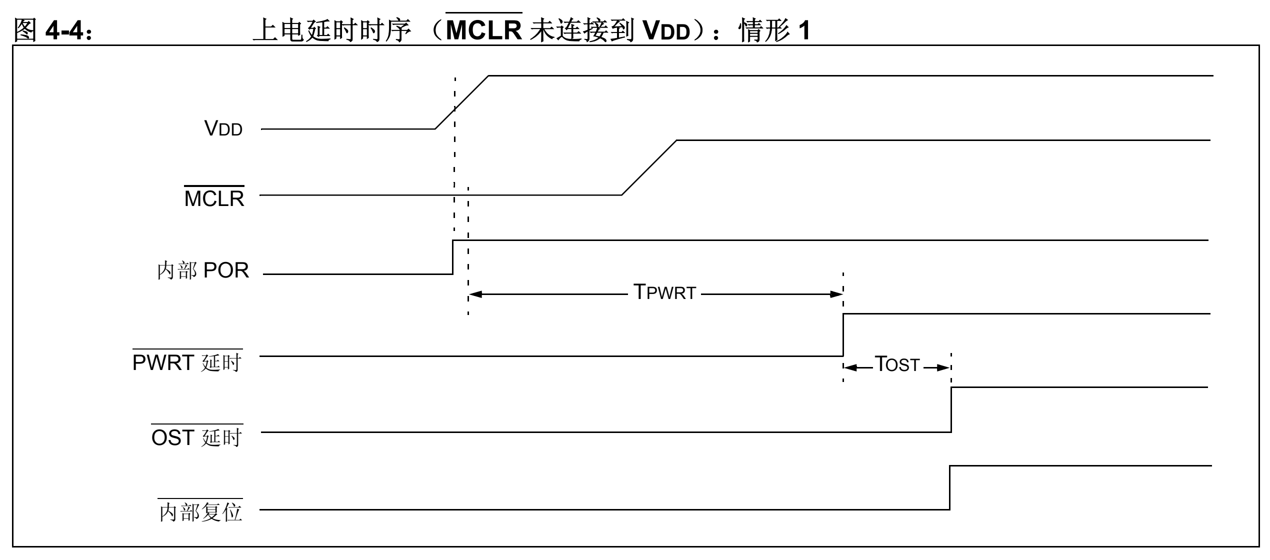


图 4-5： 上电延时时序（$\overline{\text{MCLR}}$ 未连接到 VDD）：情形 2

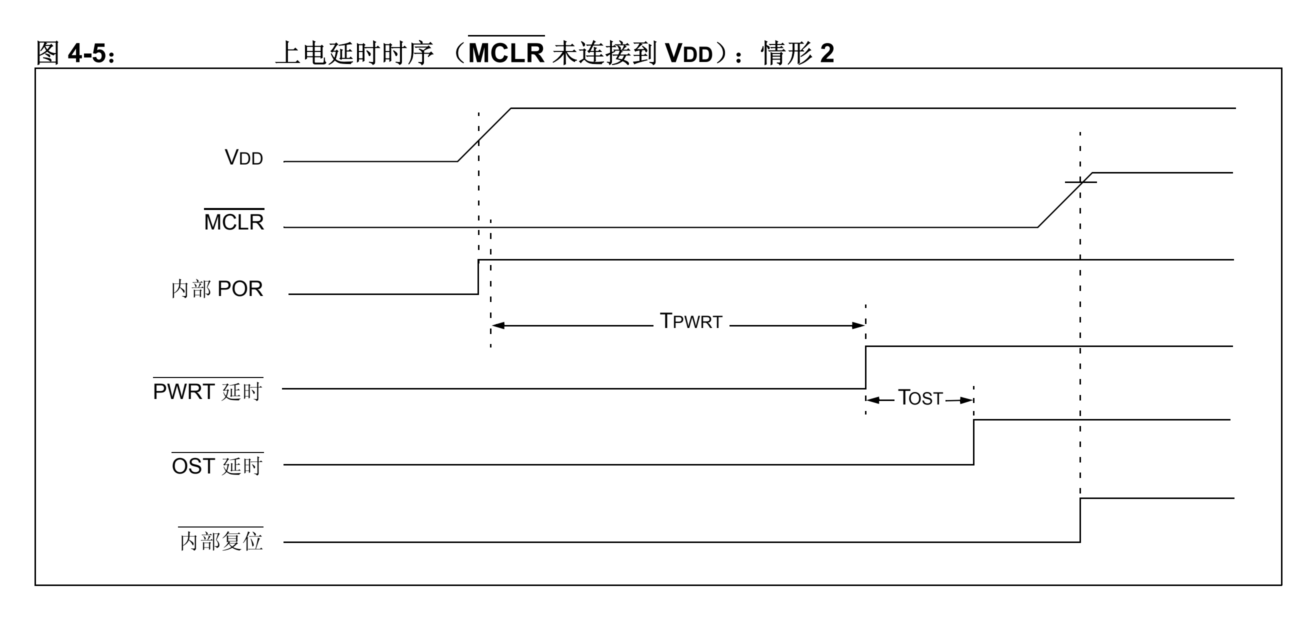


图 4-6： 缓慢上升时间 （$\overline{\text{MCLR}}$ 连接到 VDD， VDD 上升时间 > TPWRT)

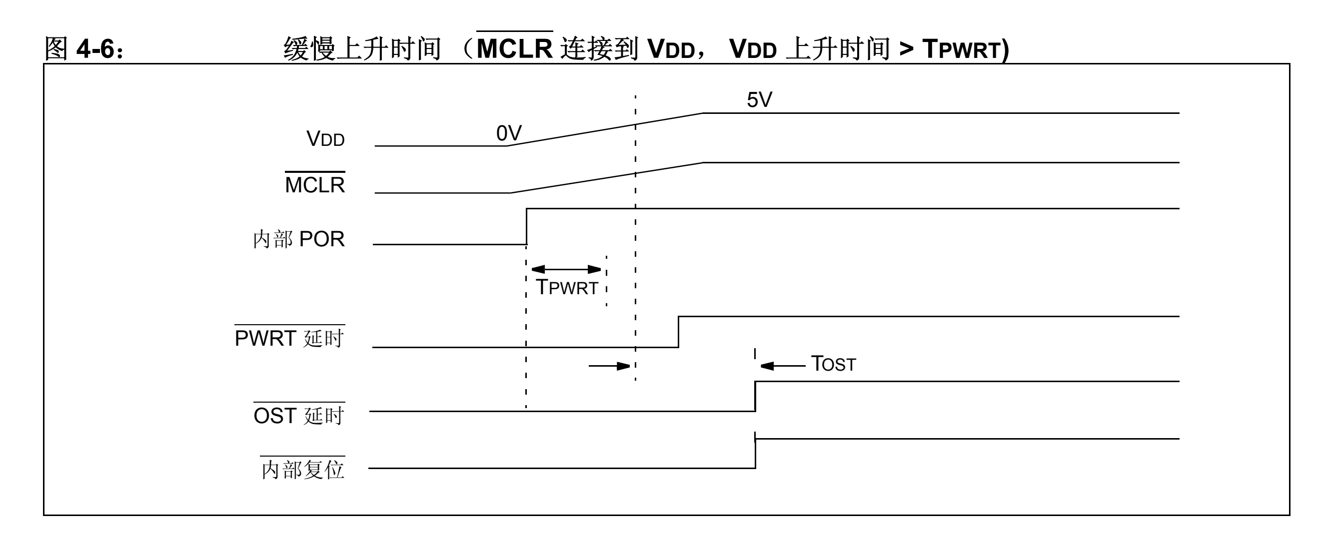
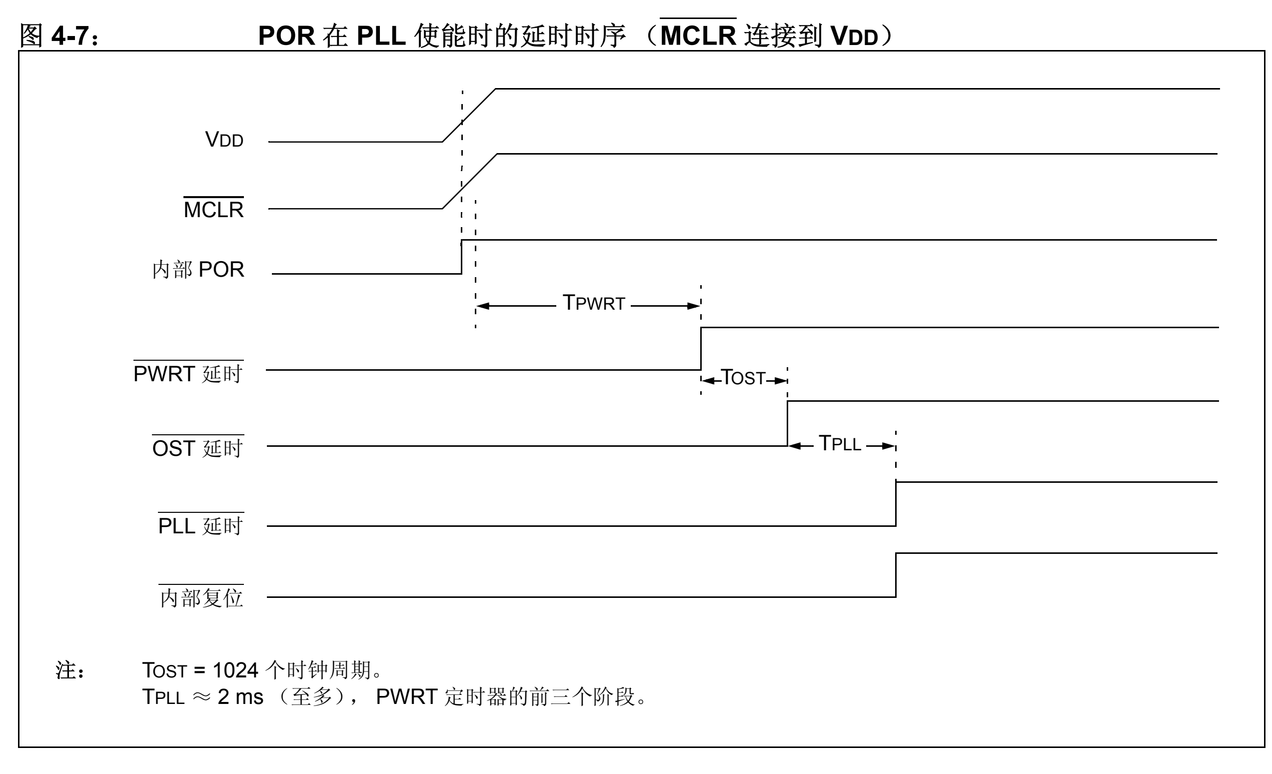


图 4-7： POR 在 PLL 使能时的延时时序 （$\overline{\text{MCLR}}$ 连接到 VDD）



注： TOST = 1024 个时钟周期。
TPLL ≈ 2 ms （至多）， PWRT 定时器的前三个阶段。

4.6 寄存器的复位状态

大多数寄存器不受复位的影响。在 POR 时这些寄存器的状态不确定，而在其他复位时它们的状态不变。而其余寄存器则根据不同的复位类型被强制为 “复位状态”。

因为 WDT 唤醒被视为恢复正常的工作，所以大部分寄存器不受 WDT 唤醒的影响。如表 4-3 所示， RCON 寄存器中的状态位：RI、TO、PD、POR 和 BOR 在不同的复位情形中会分别被置 1 或清零。可在软件中使用这些状态位判断复位的类型。

表 4-4 说明了所有特殊功能寄存器的复位状态。这些复位被分类为上电和欠压复位、主清零和 WDT 复位以及 WDT 唤醒复位。

表 4-3： RCON 寄存器的状态位、含义以及初始状态

条件	程序计数器	RCON 寄存器						STKPTR 寄存器	
		SBOREN	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
上电复位	0000h	1	1	1	1	0	0	0	0
RESET 指令	0000h	u(2)	0	u	u	u	u	u	u
欠压复位	0000h	u(2)	1	1	1	u	0	u	u
功耗管理运行模式期间的 $\overline{MCLR}$ 复位	0000h	u(2)	u	1	u	u	u	u	u
功耗管理空闲模式和休眠模式期间的 MCLR 复位	0000h	u(2)	u	1	0	u	u	u	u
全功耗或功耗管理运行模式期间的 WDT 超时	0000h	u(2)	u	0	u	u	u	u	u
全功耗执行期间的 $\overline{MCLR}$ 复位	0000h	u(2)	u	u	u	u	u	u	u
堆栈满复位 （STVREN = 1）	0000h	u(2)	u	u	u	u	u	1	u
堆栈下溢复位 （STVREN = 1）	0000h	u(2)	u	u	u	u	u	u	1
堆栈下溢错误 （不是真正的复位， STVREN = 0）	0000h	u(2)	u	u	u	u	u	u	1
功耗管理空闲或休眠模式期间的 WDT 超时	PC + 2	u(2)	u	0	0	u	u	u	u
通过中断从功耗管理模式退出	PC + 2(1)	u(2)	u	u	0	u	u	u	u

图注： u = 不变

注 1： 当器件被中断唤醒且 GIEH 或 GIEL 位置 1 时， PC 装入中断向量 （0008h 或 0018h）。

2： 当软件使能 BOR 时（BOREN1:BOREN0 配置位 = 01 且 SBOREN = 1 ）， POR 位的复位状态是 1 且其他复位不能改变该状态。否则，复位状态是 0。

超前信息

表 4-4：　所有寄存器的初始状态

寄存器	适用器件		上电复位 或欠压复位	$\overline{\text{MCLR}}$ 复位、 WDT 复位、 RESET 指令 或堆栈复位	通过 WDT 或中断唤醒器件
TOSU	1230	1330	---0 0000	---0 0000	---0 uuuu[3]
TOSH	1230	1330	0000 0000	0000 0000	uuuu uuuu[3]
TOSL	1230	1330	0000 0000	0000 0000	uuuu uuuu[3]
STKPTR	1230	1330	00-0 0000	uu-0 0000	uu-u uuuu[3]
PCLATU	1230	1330	---0 0000	---0 0000	---u uuuu
PCLATH	1230	1330	0000 0000	0000 0000	uuuu uuuu
PCL	1230	1330	0000 0000	0000 0000	PC + 2[2]
TBLPTRU	1230	1330	--00 0000	--00 0000	--uu uuuu
TBLPTRH	1230	1330	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	1230	1330	0000 0000	0000 0000	uuuu uuuu
TABLAT	1230	1330	0000 0000	0000 0000	uuuu uuuu
PRODH	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	1230	1330	0000 000x	0000 000u	uuuu uuuu[1]
INTCON2	1230	1330	1111 1111	1111 1111	uuuu uuuu[1]
INTCON3	1230	1330	1100 0000	1100 0000	uuuu uuuu[1]
INDF0	1230	1330	N/A	N/A	N/A
POSTINC0	1230	1330	N/A	N/A	N/A
POSTDEC0	1230	1330	N/A	N/A	N/A
PREINC0	1230	1330	N/A	N/A	N/A
PLUSW0	1230	1330	N/A	N/A	N/A
FSR0H	1230	1330	---- 0000	---- 0000	---- uuuu
FSR0L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	1230	1330	N/A	N/A	N/A
POSTINC1	1230	1330	N/A	N/A	N/A
POSTDEC1	1230	1330	N/A	N/A	N/A
PREINC1	1230	1330	N/A	N/A	N/A
PLUSW1	1230	1330	N/A	N/A	N/A
FSR1H	1230	1330	---- 0000	---- 0000	---- uuuu
FSR1L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
SR	1230	1330	---- 0000	---- 0000	---- uuuu

图注：　u = 不变， x = 未知， - = 未用 （读为 0），q = 取值视条件而定。

注　1：INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

2：当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3：当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一级。

4：具体条件下的复位值，请参见表 4-3。

5：根据选定的振荡器模式使能 PORTA、 LATA 和 TRISA 中的 bit 6 位和 bit 7。若未将这两位配置为 PORTA 引脚，则它们将被禁止且读为 0。

6：PWMEN 位的复位状态由 CONFIG3L 中 PWMPIN 配置位的设置决定。

表 4-4： 所有寄存器的初始状态 （续）

寄存器	适用器件		上电复位 或欠压复位	$\overline{\text{MCLR}}$ 复位、 WDT 复位、 RESET 指令 或堆栈复位	通过 WDT 或中断唤醒器件
INDF2	1230	1330	N/A	N/A	N/A
POSTINC2	1230	1330	N/A	N/A	N/A
POSTDEC2	1230	1330	N/A	N/A	N/A
PREINC2	1230	1330	N/A	N/A	N/A
PLUSW2	1230	1330	N/A	N/A	N/A
FSR2H	1230	1330	---- 0000	---- 0000	---- uuuu
SR2L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	1230	1330	---x xxxx	---u uuuu	---u uuuu
TMR0H	1230	1330	0000 0000	0000 0000	uuuu uuuu
TMR0L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	1230	1330	1111 1111	1111 1111	uuuu uuuu
OSCCON	1230	1330	0100 q000	0100 q000	uuuu uuqu
LVDCON	1230	1330	--00 0101	--00 0101	--uu uuuu
WDTCON	1230	1330	---- ---0	---- ---0	---- ---u
RCON(4)	1230	1330	0q-1 11q0	0q-q qquu	uq-u qquu
TMR1H	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	1230	1330	0000 0000	u0uu uuuu	uuuu uuuu
ADRESH	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	1230	1330	0--- 0000	0--- 0000	u--- uuuu
ADCON1	1230	1330	---0 1111	---0 1111	---u uuuu
ADCON2	1230	1330	0-00 0000	0-00 0000	u-uu uuuu
BAUDCON	1230	1330	01-0 0-00	01-0 0-00	--uu uuuu
CVRCON	1230	1330	0-00 0000	0-00 0000	u-uu uuuu
CMCON	1230	1330	000- -000	000- -000	uuu- -uuu
SPBRGH	1230	1330	0000 0000	0000 0000	uuuu uuuu
SPBRG	1230	1330	0000 0000	0000 0000	uuuu uuuu
RCREG	1230	1330	0000 0000	0000 0000	uuuu uuuu
TXREG	1230	1330	0000 0000	0000 0000	uuuu uuuu
TXSTA	1230	1330	0000 0010	0000 0010	uuuu uuuu
RCSTA	1230	1330	0000 000x	0000 000x	uuuu uuuu

图注： u = 不变， x = 未知， - = 未用 （读为 0）， q = 取值视条件而定。

注 1： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一级。

4： 具体条件下的复位值，请参见表 4-3。

5： 根据选定的振荡器模式使能 PORTA、 LATA 和 TRISA 中的 bit 6 位和 bit 7。若未将这两位配置为 PORTA 引脚，则它们将被禁止且读为 0。

6： PWMEN 位的复位状态由 CONFIG3L 中 PWMPIN 配置位的设置决定。

表 4-4：　所有寄存器的初始状态 （续）

寄存器	适用器件		上电复位 或欠压复位	MCLR 复位、 WDT 复位、 RESET 指令 或堆栈复位	通过 WDT 或中断唤醒器件
EEADR	1230	1330	0000 0000	0000 0000	uuuu uuuu
EEDATA	1230	1330	0000 0000	0000 0000	uuuu uuuu
EECON2	1230	1330	0000 0000	0000 0000	0000 0000
EECON1	1230	1330	xx-0 x000	uu-0 u000	uu-0 u000
IPR3	1230	1330	---0 ----	---0 ----	---u ----
PIR3	1230	1330	---0 ----	---0 ----	---u ----
PIE3	1230	1330	---0 ----	---0 ----	---u ----
IPIR2	1230	1330	1--1 -1--	1--1 -1--	u--u -u--
PIR2	1230	1330	0--0 -0--	0--0 -0--	u--u -u--(1)
PIE2	1230	1330	0--0 -0--	0--0 -0--	u--u -u--
IPR1	1230	1330	-111 1111	-111 1111	-uuu uuuu
PIR1	1230	1330	-000 0000	-000 0000	-uuu uuuu(1)
PIE1	1230	1330	-000 0000	-000 0000	-uuu uuuu
OSCTUNE	1230	1330	00-0 0000	00-0 0000	uu-u uuuu
PTCON0	1230	1330	0000 0000	uuuu uuuu	uuuu uuuu
PTCON1	1230	1330	00-- ----	00-- ----	uu-- ----
PTMRL	1230	1330	0000 0000	0000 0000	uuuu uuuu
PTMRH	1230	1330	---- 0000	---- 0000	---- uuuu
PTPERL	1230	1330	1111 1111	1111 1111	uuuu uuuu
PTPERH	1230	1330	---- 1111	---- 1111	---- uuuu
TRISB	1230	1330	1111 1111	1111 1111	uuuu uuuu
TRISA	1230	1330	1111 1111(5)	1111 1111(5)	uuuu uuuu(5)
PDC0L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC0H	1230	1330	--00 0000	--00 0000	--uu uuuu
PDC1L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC1H	1230	1330	--00 0000	--00 0000	--uu uuuu
PDC2L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC2H	1230	1330	--00 0000	--00 0000	--uu uuuu
FLTCONFIG	1230	1330	0--- -000	0--- -000	u--- -uuu
LATB	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	1230	1330	xxxx xxxx(5)	uuuu uuuu(5)	uuuu uuuu(5)

图注：　u = 不变， x = 未知， - = 未用 （读为 0），q = 取值视条件而定。

注　**1：** INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一级。

4： 具体条件下的复位值，请参见表 4-3。

5： 根据选定的振荡器模式使能 PORTA、 LATA 和 TRISA 中的 bit 6 位和 bit 7。若未将这两位配置为 PORTA 引脚，则它们将被禁止且读为 0。

6： PWMEN 位的复位状态由 CONFIG3L 中 PWMPIN 配置位的设置决定。

表 4-4： 所有寄存器的初始状态 （续）

寄存器	适用器件		上电复位 或欠压复位	$\overline{MCLR}$ 复位、 WDT 复位、 RESET 指令 或堆栈复位	通过 WDT 或中断唤醒器件
SEVTCMPL	1230	1330	0000 0000	0000 0000	uuuu uuuu
SEVTCMPH	1230	1330	---- 0000	---- 0000	---- uuuu
PWMCON0	1230	1330	-100 -000(6)	-100 -000(6)	-uuu -uuu(6)
			-000 -000(6)	-000 -000(6)	-uuu -uuu(6)
PWMCON1	1230	1330	0000 0-00	0000 0-00	uuuu u-uu
DTCON	1230	1330	0000 0000	0000 0000	uuuu uuuu
OVDCOND	1230	1330	--11 1111	--11 1111	--uu uuuu
OVDCONS	1230	1330	--00 0000	--00 0000	--uu uuuu
PORTB	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	1230	1330	xx0x 0000(5)	uu0u 0000(5)	uuuu uuuu(5)

图注： u = 不变， x = 未知， - = 未用 （读为 0）， q = 取值视条件而定。

注 1： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一级。

4： 具体条件下的复位值，请参见表 4-3。

5： 根据选定的振荡器模式使能 PORTA、 LATA 和 TRISA 中的 bit 6 位和 bit 7。若未将这两位配置为 PORTA 引脚，则它们将被禁止且读为 0。

6： PWMEN 位的复位状态由 CONFIG3L 中 PWMPIN 配置位的设置决定。

 超前信息

5.0 存储器构成

在 PIC18 增强型单片机器件上有三种类型的存储器：

- 程序存储器
- 数据 RAM
- 数据 EEPROM

在哈佛架构的器件中，数据和程序存储器使用不同的总线，因而可同时访问这两种存储器空间。数据 EEPROM，从实际用途而言，可以被看作外设器件，因为它是通过一组控制寄存器来寻址和访问的。

第 6.0 节 "闪存程序存储器"提供了闪存程序存储器操作的详细信息。**第 7.0 节 "数据 EEPROM 存储器"**中将单独讨论数据 EEPROM。

5.1 程序存储器构成

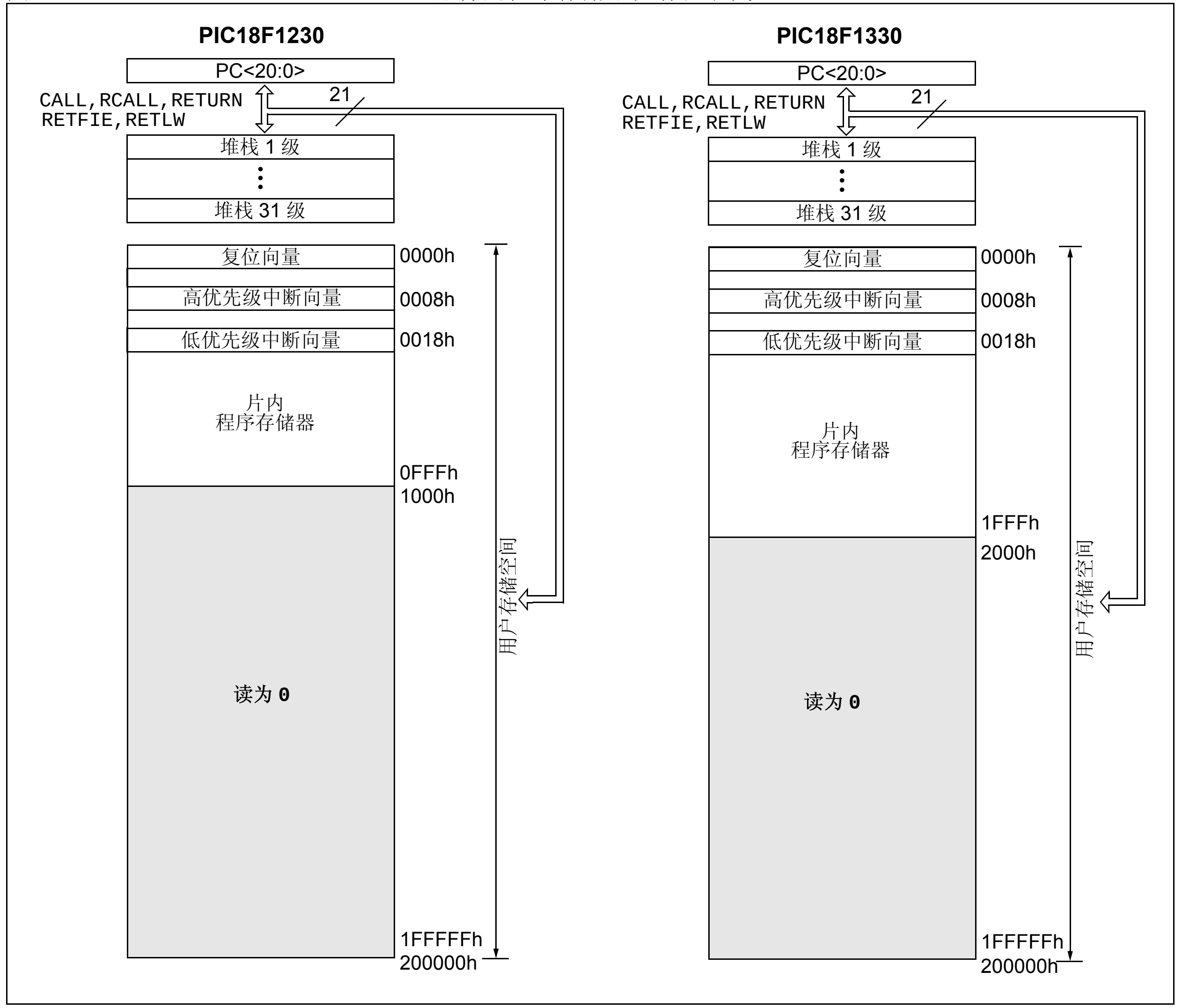
PIC18 单片机实现了 21 位程序计数器，可以对 2 MB 的程序存储器空间进行寻址。访问存储器物理地址上边界和2 MB地址之间的存储单元将会返回全0（NOP指令）。

PIC18F1230 具有 4 KB 闪存，可存储多达 2048 条单字指令。PIC18F1330 具有 8 KB 闪存，可存储多达 4096 条单字指令。

PIC18 器件有两个中断向量。复位向量地址为 0000h，中断向量地址为 0008h 和 0018h。

图 5-1 介绍了 PIC18F1230 和 PIC18F1330 器件的程序存储器映射。

图 5-1： PIC18F1230/1330 器件的程序存储器映射和堆栈



5.1.1 程序计数器

程序计数器（Program Counter，PC）指定要取出执行的指令地址。PC 为 21 位宽，并且包含在 3 个不同的 8 位寄存器中。其中的低字节称为 PCL 寄存器，该寄存器可读写。高字节，即 PCH 寄存器，包含 PC<15:8> 位，不可直接读写。可以通过 PCLATH 寄存器更新 PCH 寄存器。最高字节称为 PCU。该寄存器包含 PC<20:16> 位，也不可直接读写。可以通过 PCLATU 寄存器更新 PCU 寄存器。

通过执行写 PCL 的操作，可以将 PCLATH 和 PCLATU 的内容传送到程序计数器。类似的，通过执行读 PCL 的操作，可以将程序计数器的两个高字节传送到 PCLATH 和 PCLATU。这对于计算 PC 的偏移量很有用（见**第 5.1.4.1 节 “计算 GOTO”**）。

PC 在程序存储器中按字节寻址。为防止 PC 不能正确获取指令字，需要将 PCL 的最低有效位固定取值为 0。PC 每次加 2 来连续寻址程序存储器中的指令。

CALL、RCALL、GOTO 和程序跳转指令直接写入程序计数器。对于这些指令，PCLATH 和 PCLATU 的内容将不会被传送到程序计数器。

5.1.2 返回地址堆栈

用于存放返回地址的堆栈允许保存最多 31 个程序调用地址和中断向量。当执行 CALL 或 RCALL 指令或响应中断时，PC 值被压入堆栈。在执行 RETURN、RETLW 或 RETFIE 指令时，PC 值从堆栈弹出。PCLATU 和 PCLATH 不受 RETURN 或 CALL 指令的影响。

通过 21 位的 RAM 和 5 位的堆栈指针（STKPTR）来实现 31 级深的堆栈操作。堆栈既不占用程序存储空间也不占用数据存储空间。堆栈指针可以读写，并且通过栈顶的特殊文件寄存器可以读写栈顶地址。也可使用这些寄存器将数据压入堆栈，或将数据从堆栈弹出。

执行 CALL 类型的指令会引起压栈操作；堆栈指针首先加 1，并且将 PC 的内容写入堆栈指针指向的单元（PC 已经指向 CALL 后的指令）。执行 RETURN 类型的指令时，引起出栈操作；STKPTR 寄存器所指向的单元的内容被传送给 PC，然后堆栈指针减 1。

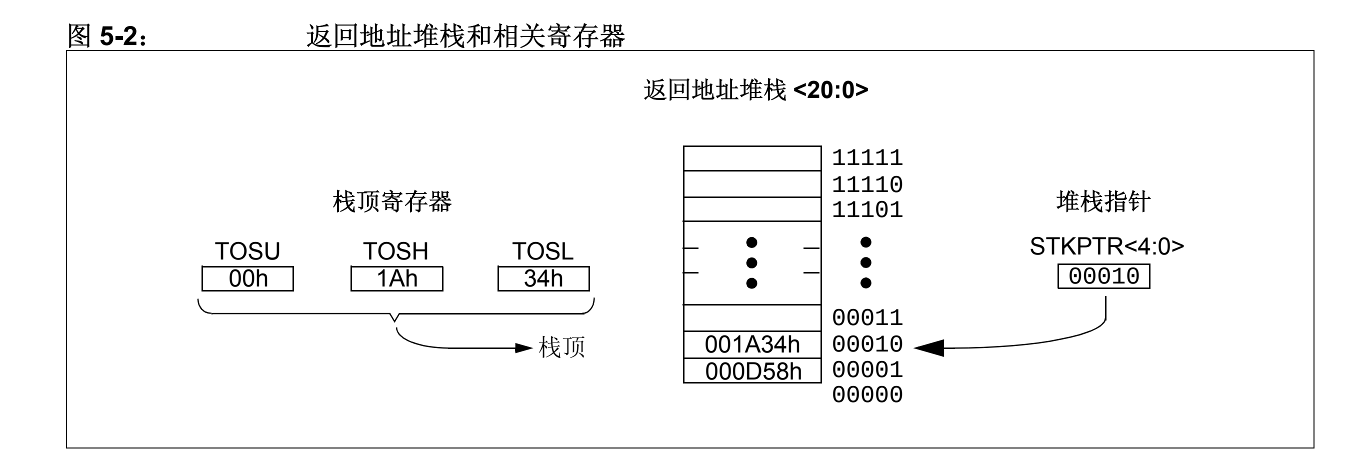
所有复位后，堆栈指针均会初始化为 00000。堆栈指针值 00000 不指向任何 RAM 单元，它仅仅是一个复位值。状态位表明堆栈是满、上溢还是下溢。

5.1.2.1 栈顶访问

只有栈顶（Top-of-Stack，TOS）是可读写的。有 3 个寄存器 TOSU:TOSH:TOSL 用于保存 STKPTR 寄存器（图 5-2）所指向的堆栈单元的内容。这可以让用户在必要时实现软件堆栈。在 CALL、RCALL 或中断后，软件可以通过读 TOSU:TOSH:TOSL 寄存器来读取压入堆栈的值。这些值可以被置入用户定义的软件堆栈。返回时，软件将这些值存回到 TOSU:TOSH:TOSL 并执行返回。

为防止对堆栈的意外操作，访问堆栈时用户必须禁止全局中断。

图 5-2： 返回地址堆栈和相关寄存器



 超前信息

5.1.2.2 返回堆栈指针 （STKPTR）

STKPTR 寄存器（寄存器 5-1）包含堆栈指针值、STKFUL（堆栈满）状态位和 STKUNF（堆栈下溢）状态位。堆栈指针值可为 0 到 31 之间的整数。向堆栈压入值前，堆栈指针加 1 ；而从堆栈弹出值后，堆栈指针减 1。复位时，堆栈指针值为零。用户可以读写堆栈指针的值。实时操作系统（Real-Time Operating System，RTOS）可以利用此特性对返回堆栈进行维护。

当向堆栈压入 PC 值 31 次（且没有值从堆栈弹出）后，STKFUL位就会置1。通过软件或POR清零STKFUL位。

堆栈满时执行的操作由 STVREN（堆栈上溢复位使能）配置位的状态决定。（有关器件配置位的介绍，请参见**第 19.1 节 "配置位"**。）如果 STVREN 位已经置 1（默认），第 31 次压栈将把 （PC + 2）值压入堆栈，将 STKFUL 位置 1，并复位器件。STKFUL 位将保持置 1，而堆栈指针将被清零。

如果 STVREN 位被清零，第 31 次进栈时 STKFUL 位会被置 1，堆栈指针则加 1 变为 31。任何其他进栈操作都不会覆盖第31次压栈的值，并且STKPTR将保持为31。

当堆栈弹出次数足够卸空堆栈时，下一次出栈会向 PC 返回一个零值，并将 STKUNF 位置 1，而堆栈指针则保持为 0。STKUNF 位将保持置 1，直到被软件清零或发生 POR。

注：	下溢会导致向 PC 返回一个零值，并使程序跳转到复位向量处，此时可以验证堆栈状态并采取相应的操作。这与复位不同，因为 SFR 的内容不受影响。

5.1.2.3 PUSH 和 POP 指令

由于栈顶是可以读写的，因此将值压入堆栈或从堆栈弹出值而不影响程序的正常执行是非常理想的。PIC18 指令集包括两条指令 PUSH 和 POP，它们允许在软件控制下对 TOS 进行操作。可以通过修改 TOSU、 TOSH 和 TOSL，将数据或返回地址压入堆栈。

PUSH指令将当前的PC值压入堆栈。先将堆栈指针加1，再将当前 PC 值装入堆栈。

POP 指令通过将堆栈指针减 1 来丢弃当前的 TOS 值。然后前一个进栈值成为 TOS 值。

寄存器 5-1： STKPTR：堆栈指针寄存器

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

图注：	C = 可清零位		
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **STKFUL**：堆栈满标志位 (1)
1 = 堆栈满或上溢
0 = 堆栈未满或未上溢

bit 6 **STKUNF**：堆栈下溢标志位 (1)
1 = 发生堆栈下溢
0 = 未发生堆栈下溢

bit 5 未用：读为 0

bit 4-0 **SP4:SP0**：堆栈指针地址位

注 1： 通过用户软件或 POR 清零 bit 7 和 bit 6。

5.1.2.4 堆栈满和下溢复位

通过将配置寄存器 4L 中的 STVREN 位置 1，允许在出现堆栈上溢和堆栈下溢条件时使器件复位。当 STVREN 位置 1 时，堆栈满或堆栈下溢条件会将相应的 STKFUL 或 STKUNF 位置 1，然后使器件复位。当 STVREN 位清零时，堆栈满或堆栈下溢条件会将相应的 STKFUL 或 STKUNF 位置 1，但不会使器件复位。通过用户软件或上电复位使 STKFUL 或 STKUNF 位清零。

5.1.3 快速寄存器堆栈

为 STATUS、WREG 和 BSR 寄存器提供的快速寄存器堆栈具有从中断 “快速返回” 的功能。每个寄存器堆栈的深度仅为 1 级，并且不可读写。当处理器转入中断向量处执行指令时，此堆栈装入对应寄存器的当前值。所有中断源都会将值压入堆栈寄存器。如果使用 RETFIE, FAST 指令从中断返回，堆栈寄存器中的值会被重新装回对应的寄存器。

如果同时使能了低优先级中断和高优先级中断，从低优先级中断返回时，无法可靠地使用堆栈寄存器。如果在为低优先级中断提供服务时，发生了高优先级中断，则低优先级中断储存在堆栈寄存器中的值将被覆盖。在这种情况下，用户必须在低优先级中断期间用软件保存关键寄存器的值。

如果未使用中断优先级，所有中断都可以使用快速寄存器堆栈从中断返回。如果没有使用中断，快速寄存器堆栈可以用于在子程序调用结束后恢复 STATUS、WREG 和 BSR 寄存器。要将快速寄存器堆栈用于子程序调用，必须执行 CALL lable,FAST 指令将 STATUS、WREG 和 BSR 寄存器的内容存入快速寄存器堆栈。在调用结束后执行 RETURN,FAST 指令，从快速寄存器堆栈中弹出并恢复这些寄存器的值。

例 5-1 给出了一个在子程序调用和返回期间使用快速寄存器堆栈的源代码示例。

例 5-1： 快速寄存器堆栈代码示例

```
CALL  SUB1, FAST    ;STATUS, WREG, BSR
                    ;SAVED IN FAST REGISTER
                    ;STACK
      •
      •

SUB1  •
      •
      RETURN, FAST  ;RESTORE VALUES SAVED
                    ;IN FAST REGISTER STACK
```

5.1.4 程序存储器中的查找表

有些编程需要在程序存储器中创建数据结构或查找表。对于 PIC18 器件，有两种方法可以实现查找表：

- 计算 GOTO
- 表读

5.1.4.1 计算 GOTO

计算 GOTO 是通过向程序计数器加一个偏移量来实现的。例 5-2 给出了一个示例。

使用 ADDWF PCL 指令和一组 RETLW nn 指令可以创建一个查找表。在调用该表前，会先将查找表的偏移量装入 W 寄存器。被调用程序的第一条指令是 ADDWF PCL 指令。接下去执行的是一条 RETLW nn 指令，它将数值 nn 返回给调用函数。

偏移量 （WREG 中的值）指定程序计数器应该增加的字节数，其值应当为 2 的倍数 （LSb = 0）。

在这种方法中，每个指令单元只能存储一个数据字节，并且要求返回地址堆栈还有空闲单元。

例 5-2： 使用偏移量计算 GOTO

```
       MOVF    OFFSET, W
       CALL    TABLE
ORG    nn00h
TABLE  ADDWF   PCL
       RETLW   nnh
       RETLW   nnh
       RETLW   nnh
       .
       .
       .
```

5.1.4.2 表读和表写

有一种更好的方法可以将数据存储在程序存储器中，该方法允许在每个指令单元存储 2 个字节的数据。

使用表读和表写，每个程序字可以存储 2 个字节的查找表数据。表指针寄存器 （TBLPTR）指定字节地址，而表锁存器 （TABLAT）储存从程序存储器读取或写入程序存储器的数据。进出程序存储器的数据每次为一个字节。

第 6.1 节 “表读和表写”中将进一步讨论表读和表写操作。

5.2 PIC18 指令周期

5.2.1 时钟分配

单片机时钟输入信号，无论来自内部或外部时钟源，都会在器件内部被 4 分频用来产生 4 个不重叠的正交时钟信号，即 Q1、 Q2、 Q3 和 Q4。程序计数器在每个 Q1 递增，并在 Q4 期间从程序存储器取指并将指令锁存到指令寄存器中。指令的译码和执行在下一个 Q1 到 Q4 周期完成。图 5-3 所示为时钟和指令执行的流程图。

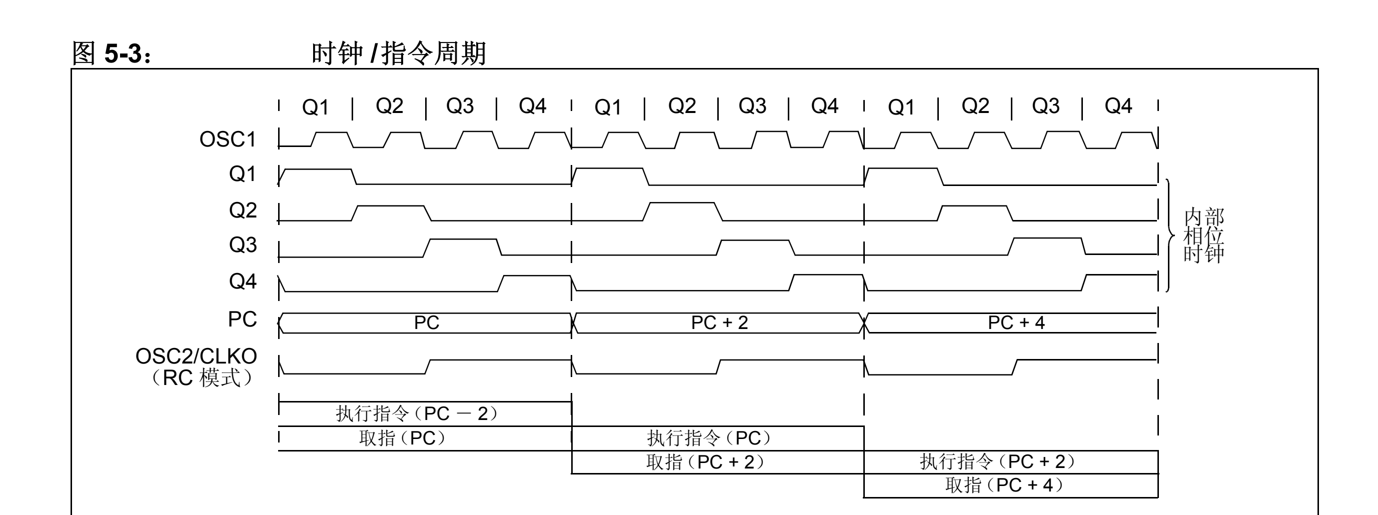
5.2.2 指令流 / 流水线

一个 “指令周期”由 4 个 Q 周期组成，即 Q1 到 Q4。指令的取指和执行是以流水线的形式进行的，用一个指令周期来取指，而用另一个指令周期译码和执行指令。但由于是流水线操作，所以每条指令的等效执行时间都是一个指令周期。如果某条指令改变了程序计数器（如 GOTO 指令），则需要两个指令周期才能完成该指令（见例 5-3）。

取指周期：程序计数器（PC）在 Q1 周期递增，开始取指。

指令执行周期：在 Q1 周期，将所取指令锁存到指令寄存器（Instruction Register， IR）。然后在 Q2、 Q3 和 Q4 周期中进行指令的译码和执行。其中读数据存储器（读操作数）发生在 Q2 周期，写操作发生在 Q4 周期（写目标地址）。

图 5-3： 时钟 / 指令周期



例 5-3： 指令流水线流程

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	取指 1	执行 1				
2. MOVWF PORTB		取指 2	执行 2			
3. BRA SUB_1			取指 3	执行 3		
4. BSF PORTA, BIT3（强制 NOP）				取指 4	清除 （NOP）	
5. 在 SUB_1 地址处的指令					取指 SUB_1	执行 SUB_1

除程序跳转指令之外，所有指令都是单周期指令。由于程序跳转指令将导致一条已取指令从流水线清除，需要重新取指，然后执行指令，所以程序跳转指令需要两个周期。

5.2.3 程序存储器中的指令

程序存储器按字节寻址。指令以 2 字节或 4 字节形式存储在程序存储器中。指令字的最低有效字节始终存储在地址为偶数的程序存储器单元中（LSB = 0）。要保证正确指向指令单元，PC 必须以 2 为单位递增，并且 LSB 总是 0 （见**第 5.1.1 节 “程序计数器”**）。

图 5-4 给出了指令字存储在程序存储器中的一个示例。

CALL 和 GOTO 指令在指令中嵌入了程序存储器的绝对地址。指令总是存储为一个字长，因而指令所包含的数据为字地址。字地址会写入 PC<20:1>，由 PC 在程序存储器中访问目标地址。图 5-4中的指令2给出了指令“GOTO 0006h”在程序存储器中的译码过程。程序跳转指令也采取同样的方式对相对地址偏移量进行译码。在跳转指令中的偏移量代表单字指令数，PC将以此作为偏移量跳转到指定的地址单元。**第 21.0 节 “指令集综述”**提供了指令集的更多详情。

图 5-4： 程序存储器中的指令

程序存储器字节单元 →			LSB = 1	LSB = 0	字地址 ↓
					000000h
					000002h
					000004h
					000006h
指令 1：	MOVLW	055h	0Fh	55h	000008h
指令 2：	GOTO	0006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
指令 3：	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

5.2.4 双字指令

标准的 PIC18 指令集有 4 条双字指令：CALL、MOVFF、GOTO 和 LSFR。在所有情况下，这些指令第二个字的 4 个最高有效位总是 1111，而其余 12 位是立即数数据，通常为数据存储器地址。

指令的 4 个最高有效位 1111 用于指定一个特殊的 NOP 指令。指令的正确执行顺序为：执行完第一个字之后立即按顺序访问并使用第二个字中的数据。如果由于某些原因跳过了第一个字并自行执行指令的第二个字，那么将执行 NOP 指令。如果双字指令跟在更改 PC 的条件指令后，就有必要执行此操作。例 5-4 说明了其执行过程。

> **注：** 欲知扩展指令集中的双字指令信息，请参见**第 5.6 节 “PIC18 指令执行和扩展的指令集”**。

例 5-4： 双字指令

情形 1：

目标代码	源代码
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, skip this word
1111 0100 0101 0110	; Execute this word as a NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code

情形 2：

目标代码	源代码
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; Yes, execute this word
1111 0100 0101 0110	; 2nd word of instruction
0010 0100 0000 0000	ADDWF REG3 ; continue code

5.3 数据存储器构成

注：	当使能 PIC18 扩展指令集时，数据存储器某些方面的操作会有所改变。如需更多信息，请参见**第 5.5 节 “数据存储器和扩展指令集”**。

PIC18 器件中的数据存储器是用静态 RAM 实现的。在数据存储器中，每个寄存器有 12 位地址，数据存储容量可达 4096 个字节。存储空间被分为 16 个存储区，每个存储区包含 256 个字节； PIC18F1230/1330 器件使用 1 个存储区。图 5-5 显示了 PIC18F1230/1330 器件的数据存储器构成。

数据存储器由特殊功能寄存器（Special Function Register，SFR）和通用寄存器（General Purpose Register，GPR）组成。SFR 用于单片机和外设功能模块的控制和状态显示，而 GPR 则用于在用户应用程序中存储数据和临时存储操作的中间结果。任何未用单元的读取值均为 0。

此指令集和架构支持跨存储区的操作。可以通过直接、间接或变址寻址模式访问整个数据存储器。本章后面的部分将讨论寻址模式。

为了确保能在一个周期内存取常用寄存器（SFR 和所选的 GPR），PIC18 器件设置了快速操作存储区。这是一个 256 字节的存储器空间，它可实现对 SFR 和 GPR Bank 0 的低地址单元的快速存取，而无需使用 BSR。**第5.3.2节“快速操作存储区”**提供了对于快速操作RAM 的详细说明。

5.3.1 存储区选择寄存器 （BSR）

存储容量较大的数据存储器需要有效的寻址机制，以便对所有地址进行快速存取。理想状况下，这意味着不必要为每次读写操作提供整个地址。PIC18 器件使用 RAM 区存储机制实现快速存取。该机制将存储空间分成连续的 16 个 256 字节的存储区。根据不同的指令，可以通过完整的 12 位地址直接寻址每个单元，或通过 8 位低字节地址和 4 位存储区指针间接寻址每个单元。

PIC18 指令集中的大部分指令都使用存储区指针，也就是存储区选择寄存器（Bank Select Register，BSR）。BSR 保存单元地址的 4 个最高有效位，而指令本身则包括单元地址的 8 个 最低有效位。只使用 BSR 的低四位（BSR3:BSR0）而不使用高四位，高四位的读取值始终为 0 且不能被写入。可以通过使用 MOVLB 指令直接装入 BSR。

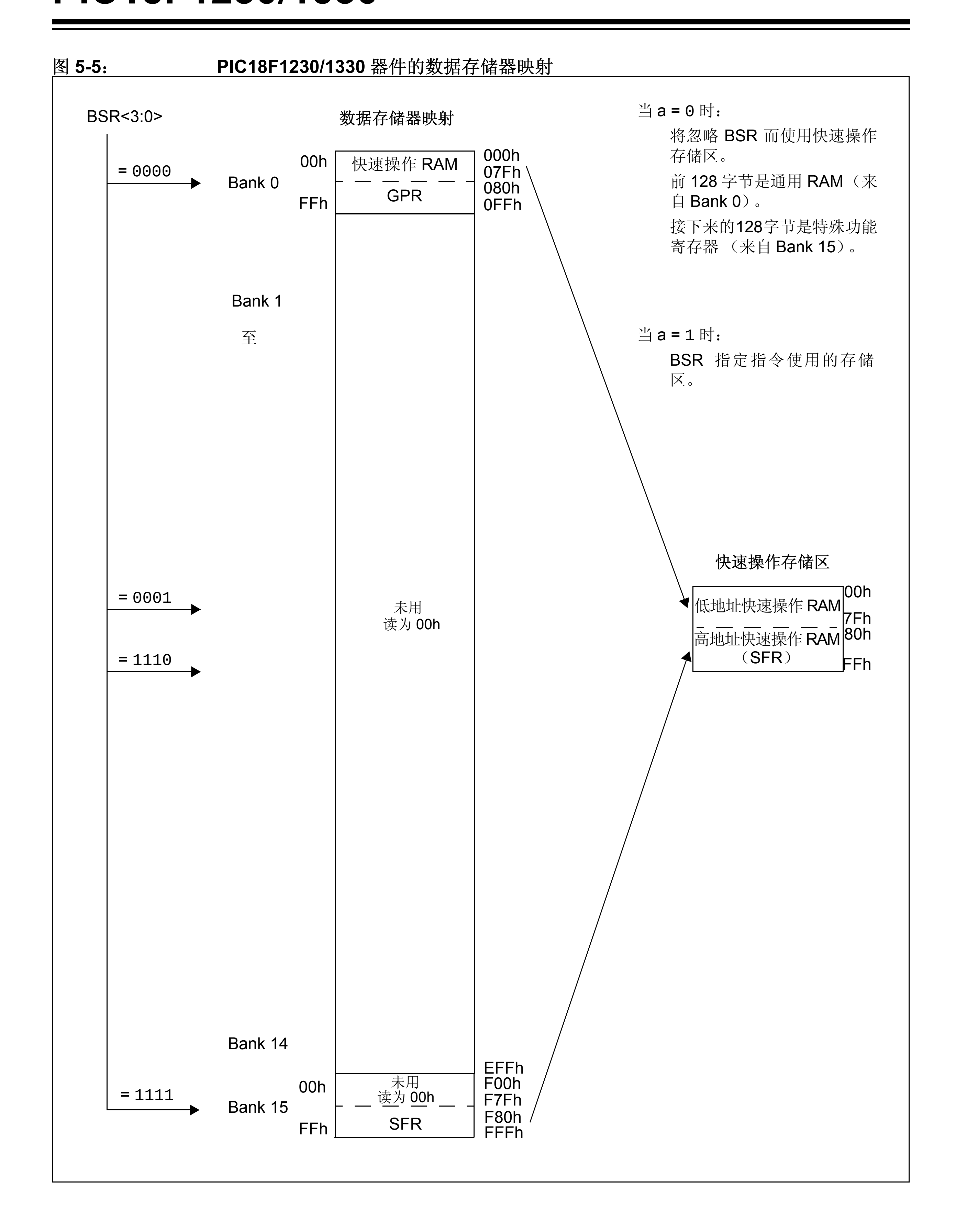
BSR 的值指定数据存储器中的存储区。指令中的 8 位指向指定存储区中的存储单元，可以将它看作是以存储区下边界为起点的偏移量。图 5-6 所示是 BSR 的值与存储区之间的关系。

由于最多有 16 个寄存器共享同一个低位地址，用户必须非常小心以确保在执行数据读或写之前选择了正确的存储区。例如，当 BSR 为 0Fh 时将程序数据写入 8 位地址 F9h 将导致程序计数器复位。

当选择存储区时，只有实际可使用的存储区可以被读写。对未用的存储区的写入将被忽略，而读未用的存储区会返回 0。虽然是这样， STATUS 寄存器仍然会受到影响。图 5-5 中的数据存储器映射图指出了可使用的存储区。

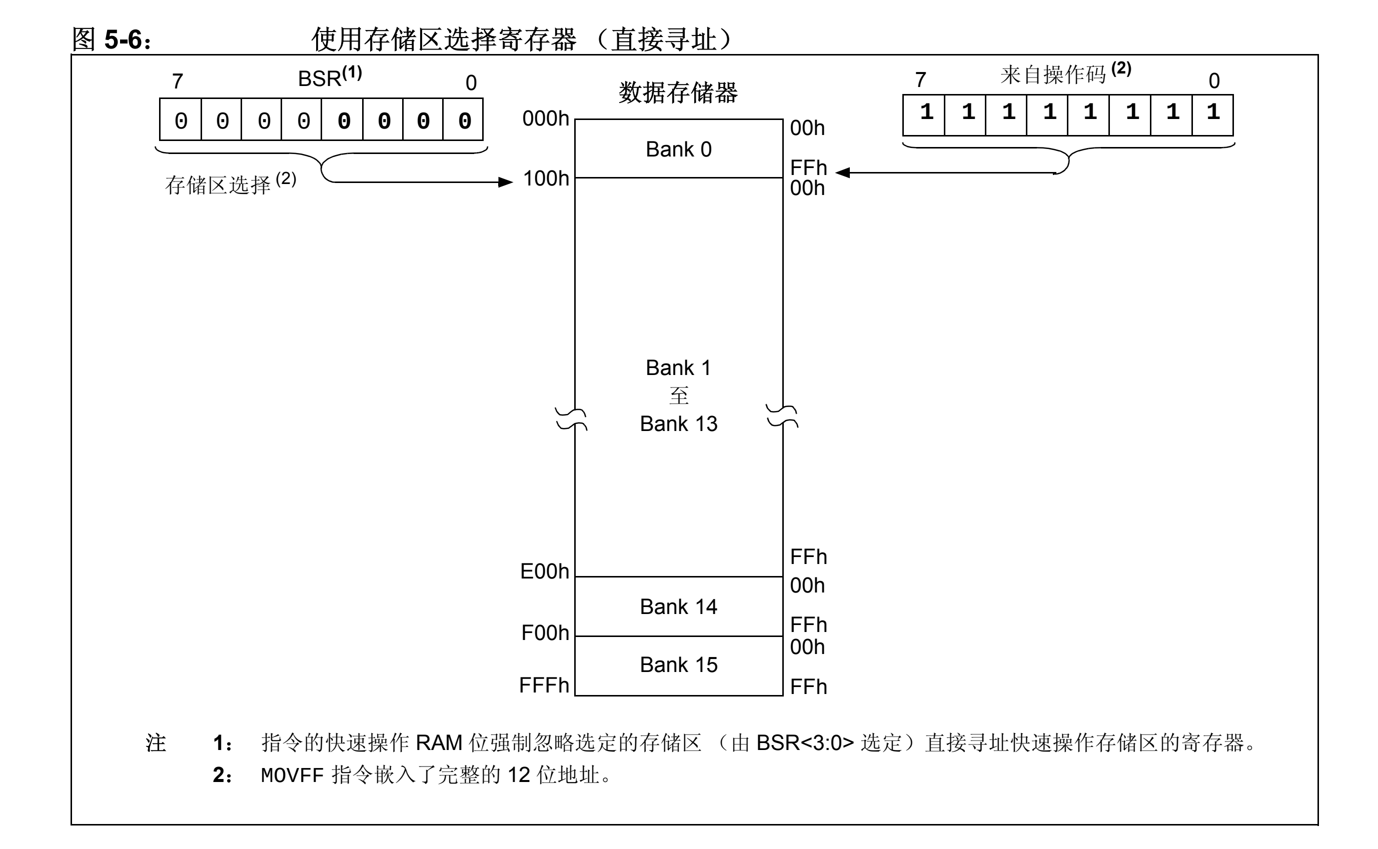
在 PIC18 的内核指令集中，只有 MOVFF 指令指定源寄存器和目标寄存器的完整 12 位地址。此指令在执行时完全忽略 BSR。所有其他指令仅包含作为操作数的低位地址，而且必须使用 BSR 或快速操作存储区来寻址目标寄存器。

图 5-5： PIC18F1230/1330 器件的数据存储器映射



 超前信息

图 5-6： 使用存储区选择寄存器 （直接寻址）



注 1： 指令的快速操作 RAM 位强制忽略选定的存储区 （由 BSR<3:0> 选定）直接寻址快速操作存储区的寄存器。
2： MOVFF 指令嵌入了完整的 12 位地址。

5.3.2 快速操作存储区

使用 BSR 和指令内嵌的 8 位地址可以使用户寻址整个数据存储空间，这同时意味着用户必须始终确保选择了正确的存储区。否则，可能会从错误的单元读取数据或将数据写入错误的单元。如果本来是向 GPR 进行写操作，却将结果写入了 SFR，后果是非常严重的。但是每次读写数据存储器时，需要验证和 / 或更改 BSR 可能会降低代码的执行效率。

为了连续访问大多数常用的数据存储单元，可以为数据存储器配置快速操作存储区，这使得用户无需指定 BSR 即可访问被映射的存储区。快速操作存储区由 Bank 0 的前 128 个字节（00h-7Fh）和 Bank 15 的后 128 个字节（80h-FFh）组成。低半部分就是 “快速操作 RAM”，由 GPR 组成。高地址的那一半被映射为器件的 SFR。这两个区域可以在快速操作存储区中连续映射并且可以用 8 位地址进行线性寻址 （图 5-5）。

快速操作存储区供包括快速操作 RAM 位 （指令中的 “a”参数）的 PIC18 内核指令使用。当 “a”等于 1 时，指令使用 BSR 和包含在操作码中的 8 位地址来对数据存储器进行寻址。但是当 “a”为 0 时，指令被强制使用快速操作存储区地址映射； BSR 的当前值被忽略。

该 “强制”寻址方式可使指令在一个周期内对数据地址进行操作，而无需首先更新 BSR。这意味着用户可以更有效地对 8 位地址为 80h 及以上的 SFR 进行取值和操作。地址低于 80h 的快速操作 RAM 非常适合于存储那些用户可能需要快速存取的数据值（如直接计算结果或常用程序变量）。快速操作 RAM 还可以实现更快速、代码效率更高的现场保护和变量切换。

当使能扩展的指令集 （XINST 配置位 = 1）时，快速操作存储区的映射略有不同。**第 5.5.3 节 “在立即数变址寻址模式下映射快速操作存储区”**更详细地讨论了此操作。

5.3.3 通用寄存器

PIC18 器件可能在 GRP 区中划分了一部分存储区。这部分存储区为数据 RAM，所有指令都可以访问它。GPR 区从 Bank 0 的底部 （地址 000h）开始向上直到 SFR 区的底部。上电复位不会将 GPR 初始化，并且其他复位也不会改变其内容。

5.3.4 特殊功能寄存器

特殊功能寄存器（SFR）是 CPU 和外设模块用来控制器件操作的寄存器。这类寄存器以静态 RAM 的形式实现。SFR 起始于数据存储器的顶部（FFFh）并向下覆盖 Bank 15 的上半部分（F80h 至 FFFh）。表 5-1 和表 5-2 列出了这些寄存器。

SFR 可分为两类：一类与“内核”器件功能（ALU、复位和中断）有关，另一类与外设功能有关。在相关的章节中将对复位和中断寄存器进行说明，而本章后面的部分将对 ALU 状态寄存器进行说明。与外设功能部件的操作相关的寄存器在外设的章节中进行说明。

SFR 通常分布在受其控制的外设中。未实现的 SFR 单元不可用，读为 0。

表 5-1： PIC18F1230/1330 器件的特殊功能寄存器映射

地址	名称	地址	名称	地址	名称	地址	名称
FFFh	TOSU	FDFh	INDF2[1]	FBFh	—[2]	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[1]	FBEh	—[2]	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[1]	FBDh	—[2]	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[1]	FBCh	—[2]	F9Ch	—[2]
FFBh	PCLATU	FDBh	PLUSW2[1]	FBBh	—[2]	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	—[2]	F9Ah	PTCON0
FF9h	PCL	FD9h	FSR2L	FB9h	—[2]	F99h	PTCON1
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	PTMRL
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—[2]	F97h	PTMRH
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—[2]	F96h	PTPERL
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	PTPERH
FF4h	PRODH	FD4h	—[2]	FB4h	CMCON	F94h	—[2]
FF3h	PRODL	FD3h	OSCCON	FB3h	—[2]	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	—[2]	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	—[2]	F91h	PDC0L
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	PDC0H
FEFh	INDF0[1]	FCFh	TMR1H	FAFh	SPBRG	F8Fh	PDC1L
FEEh	POSTINC0[1]	FCEh	TMR1L	FAEh	RCREG	F8Eh	PDC1H
FEDh	POSTDEC0[1]	FCDh	T1CON	FADh	TXREG	F8Dh	PDC2L
FECh	PREINC0[1]	FCCh	—[2]	FACh	TXSTA	F8Ch	PDC2H
FEBh	PLUSW0[1]	FCBh	—[2]	FABh	RCSTA	F8Bh	FLTCONFIG
FEAh	FSR0H	FCAh	—[2]	FAAh	—[2]	F8Ah	LATB
FE9h	FSR0L	FC9h	—[2]	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	—[2]	FA8h	EEDATA	F88h	SEVTCMPL
FE7h	INDF1[1]	FC7h	—[2]	FA7h	EECON2[1]	F87h	SEVTCMPH
FE6h	POSTINC1[1]	FC6h	—[2]	FA6h	EECON1	F86h	PWMCON0
FE5h	POSTDEC1[1]	FC5h	—[2]	FA5h	IPR3	F85h	PWMCON1
FE4h	PREINC1[1]	FC4h	ADRESH	FA4h	PIR3	F84h	DTCON
FE3h	PLUSW1[1]	FC3h	ADRESL	FA3h	PIE3	F83h	OVDCOND
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	OVDCONS
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

注 1： 这不是物理寄存器。
2： 未实现的寄存器，读为 0。

 超前信息

表 5-2: PIC18F1230/1330 寄存器汇总

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 与 BOR 时的值	详情请见：（页）
TOSU	—	—	—	栈顶最高字节 （TOS<20:16>）					---0 0000	41, 46
TOSH	栈顶高字节 （TOS<15:8>）								0000 0000	41, 46
TOSL	栈顶低字节 （TOS<7:0>）								0000 0000	41, 46
STKPTR	STKFUL(5)	STKUNF(5)	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	41, 47
PCLATU	—	—	—	PC<20:16> 的保持寄存器					---0 0000	41, 46
PCLATH	PC<15:8> 的保持寄存器								0000 0000	41, 46
PCL	PC 低字节 （PC<7:0>）								0000 0000	41, 46
TBLPTRU	—	—	bit 21	程序存储器表指针最高字节 （TBLPTR<20:16>）					--00 0000	41, 68
TBLPTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								0000 0000	41, 68
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								0000 0000	41, 68
TABLAT	程序存储器表锁存器								0000 0000	41, 68
PRODH	乘积寄存器高字节								xxxx xxxx	41, 79
PRODL	乘积寄存器低字节								xxxx xxxx	41, 79
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	41, 89
INTCON2	$\overline{RBPU}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	41, 90
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	41, 91
INDF0	使用 FSR0 的内容来寻址数据存储器——FSR0 的值不改变 （不是物理寄存器）								N/A	41, 60
POSTINC0	使用 FSR0 的内容来寻址数据存储器——FSR0 的值后增 （不是物理寄存器）								N/A	41, 60
POSTDEC0	使用 FSR0 的内容来寻址数据存储器——FSR0 的值后减 （不是物理寄存器）								N/A	41, 60
PREINC0	使用 FSR0 的内容来寻址数据存储器——FSR0 的值预增 （不是物理寄存器）								N/A	41, 60
PLUSW0	使用 FSR0 的内容来寻址数据存储器——FSR0 的值预增 （不是物理寄存器）， FSR0 的偏移量由 W 寄存器提供								N/A	41, 60
FSR0H	—	—	—	—	间接数据存储器地址指针 0 的高字节				---- 0000	41, 60
FSR0L	间接数据存储器地址指针 0 的低字节								xxxx xxxx	41, 60
WREG	工作寄存器								xxxx xxxx	41, 48
INDF1	使用 FSR1 的内容来寻址数据存储器——FSR1 的值不改变 （不是物理寄存器）								N/A	41, 60
POSTINC1	使用 FSR1 的内容来寻址数据存储器——FSR1 的值后增 （不是物理寄存器）								N/A	41, 60
POSTDEC1	使用 FSR1 的内容来寻址数据存储器——FSR1 的值后减 （不是物理寄存器）								N/A	41, 60
PREINC1	使用 FSR1 的内容来寻址数据存储器——FSR1 的值预增 （不是物理寄存器）								N/A	41, 60
PLUSW1	使用 FSR1 的内容来寻址数据存储器——FSR1 的值预增 （不是物理寄存器）， FSR1 的偏移量由 W 寄存器提供								N/A	41, 60
FSR1H	—	—	—	—	间接数据存储器地址指针 1 的高字节				---- 0000	41, 60
FSR1L	间接数据存储器地址指针 1 的低字节								xxxx xxxx	41, 60
BSR	—	—	—	—	存储区选择寄存器				---- 0000	41, 51
INDF2	使用 FSR2 的内容来寻址数据存储器——FSR2 的值不改变 （不是物理寄存器）								N/A	42, 60
POSTINC2	使用 FSR2 的内容来寻址数据存储器——FSR2 的值后增 （不是物理寄存器）								N/A	42, 60
POSTDEC2	使用 FSR2 的内容来寻址数据存储器——FSR2 的值后减 （不是物理寄存器）								N/A	42, 60
PREINC2	使用 FSR2 的内容来寻址数据存储器——FSR2 的值预增 （不是物理寄存器）								N/A	42, 60
PLUSW2	使用 FSR2 的内容来寻址数据存储器——FSR2 的值预增 （不是物理寄存器）， FSR2 的偏移量由 W 寄存器提供								N/A	42, 60
FSR2H	—	—	—	—	间接数据存储器地址指针 2 的高字节				---- 0000	42, 60
FSR2L	间接数据存储器地址指针 2 的低字节								xxxx xxxx	42, 60

图注： x = 未知， u = 不变， — = 未用， q = 取值视条件而定

注 1： 只有当 BOREN1:BOREN0 配置位 = 01 时， SBOREN 位才可用；否则它将被禁止并读为 0。请参见**第 4.4 节 “欠压复位 （BOR）”**。

2： 只有在特定振荡器配置中才可使用 PLLEN 位，否则，它被禁止并读为 0。请参见**第 2.6.4 节 “INTOSC 模式下的 PLL”**。

3： 只有当禁止主清零复位 （MCLRE 配置位 = 0）时 RA5 位才可用；否则， RA5 读为 0。该位是只读位。

4： 根据不同的主振荡器模式，可单独将 RA6/RA7 及其相关的锁存和方向位配置为端口引脚。当禁止时，这些位读为 0。

5： 通过用户软件或上电复位清零 bit 7 和 bit 6。

6： PWMEN 位的复位状态取决于 CONFIG3L 中 PWMPIN 配置位的设置。

7： 当 WDTEN 配置位使能时该位不起作用。

表 5-2: PIC18F1230/1330 寄存器汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 与 BOR 时的值	详情请见：（页）
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	42, 58
TMR0H	Timer0 寄存器的高字节								0000 0000	42, 103
TMR0L	Timer0 寄存器的低字节								xxxx xxxx	42, 103
T0CON	TMR0ON	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	42, 101
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0100 q000	42, 22
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	42, 179
WDTCON	—	—	—	—	—	—	—	SWDTEN(7)	---- ---0	42, 195
RCON	IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0q-1 11q0	42, 34
TMR1H	Timer1 寄存器的高字节								xxxx xxxx	42, 109
TMR1L	Timer1 寄存器的低字节								xxxx xxxx	42, 109
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0000 0000	42, 105
ADRESH	A/D 结果寄存器的高字节								xxxx xxxx	42, 172
ADRESL	A/D 结果寄存器的低字节								xxxx xxxx	42, 172
ADCON0	SEVTEN	—	—	—	CHS1	CHS0	GO/$\overline{DONE}$	ADON	0--- 0000	42, 163
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	---0 1111	42, 164
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	42, 165
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	42, 144
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0-00 0000	42, 177
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	000- -000	42, 173
SPBRGH	EUSART 波特率发生器寄存器的高字节								0000 0000	42, 146
SPBRG	EUSART 波特率发生器寄存器的低字节								0000 0000	42, 146
RCREG	EUSART 接收寄存器								0000 0000	42, 153
TXREG	EUSART 发送寄存器								0000 0000	42, 151
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	42, 142
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	42, 143
EEADR	EEPROM 地址寄存器								0000 0000	43, 75
EEDATA	EEPROM 数据寄存器								0000 0000	43, 75
EECON2	EEPROM 控制寄存器 2 （非物理寄存器）								0000 0000	43, 66
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	42, 67
IPR3	—	—	—	PTIP	—	—	—	—	---1 ----	43, 97
PIR3	—	—	—	PTIF	—	—	—	—	---0 ----	43, 93
PIE3	—	—	—	PTIE	—	—	—	—	---0 ----	43, 95
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	1--1 -1--	43, 97
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	0--0 -0--	43, 93
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	0--0 -0--	43, 95
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	-111 1111	43, 96
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	-000 0000	43, 92
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	-000 0000	43, 94
OSCTUNE	INTSRC	PLLEN(2)	—	TUN4	TUN3	TUN2	TUN1	TUN0	00-0 0000	43, 19
PTCON0	PTOPS3	PTOPS2	—	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	0000 0000	43, 116
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	00-- ----	43, 116

图注： x = 未知， u = 不变， — = 未用， q = 取值视条件而定

注 1： 只有当 BOREN1:BOREN0 配置位 = 01 时， SBOREN 位才可用；否则它将被禁止并读为 0。请参见**第 4.4 节 “欠压复位 （BOR）”**。

2： 只有在特定振荡器配置中才可使用 PLLEN 位，否则，它被禁止并读为 0。请参见**第 2.6.4 节 “INTOSC 模式下的 PLL”**。

3： 只有当禁止主清零复位 （MCLRE 配置位 = 0）时 RA5 位才可用；否则， RA5 读为 0。该位是只读位。

4： 根据不同的主振荡器模式，可单独将 RA6/RA7 及其相关的锁存和方向位配置为端口引脚。当禁止时，这些位读为 0。

5： 通过用户软件或上电复位清零 bit 7 和 bit 6。

6： PWMEN 位的复位状态取决于 CONFIG3L 中 PWMPIN 配置位的设置。

7： 当 WDTEN 配置位使能时该位不起作用。

表 5-2： PIC18F1230/1330 寄存器汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 与 BOR 时的值	详情请见：（页）
PTMRL	PWM 时基寄存器 （低 8 位）								0000 0000	43, 119
PTMRH	—	—	—	—	PWM 时基寄存器 （高 4 位）				---- 0000	43, 119
PTPERL	PWM 时基周期寄存器 （低 8 位）								1111 1111	43, 119
PTPERH	—	—	—	—	PWM 时基周期寄存器 （高 4 位）				---- 1111	43, 119
TRISB	PORTB 数据方向控制寄存器								1111 1111	43, 84
TRISA	TRISA7[4]	TRISA6[4]	PORTA 数据方向控制寄存器						1111 1111	43, 81
PDC0L	PWM 占空比寄存器 0 （低 8 位）								0000 0000	43, 125
PDC0H	—	—	PWM 占空比寄存器 0 （高 6 位）						--00 0000	43, 125
PDC1L	PWM 占空比寄存器 1 （低 8 位）								0000 0000	43, 125
PDC1H	—	—	PWM 占空比寄存器 1 （高 6 位）						--00 0000	43, 125
PDC2L	PWM 占空比寄存器 2 （低 8 位）								0000 0000	43, 125
PDC2H	—	—	PWM 占空比寄存器 2 （高 6 位）						--00 0000	43, 125
FLTCONFIG	BRFEN	—	—	—	—	FLTAS	FLTAMOD	FLTAEN	0--- -000	43, 137
LATB	PORTB 数据锁存器 （读取和写入数据锁存器）								xxxx xxxx	43, 84
LATA	LATA7[4]	LATA6[4]	PORTA 数据锁存器 （读取和写入数据锁存器）						xxxx xxxx	43, 81
SEVTCMPL	PWM 特殊事件比较寄存器 （低 8 位）								0000 0000	44, 138
SEVTCMPH	—	—	—	—	PWM 特殊事件比较寄存器 （高 4 位）				---- 0000	44, 138
PWMCON0	—	PWMEN2[6]	PWMEN1[6]	PWMEN0[6]	—	PMOD2	PMOD1	PMOD0	-100 -000 -000 -000	44, 117
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	0000 0-00	44, 118
DTCON	DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0	0000 0000	44, 130
OVDCOND	—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	--11 1111	44, 134
OVDCONS	—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	--00 0000	44, 134
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	44, 84
PORTA	RA7[4]	RA6[4]	RA5[3]	RA4	RA3	RA2	RA1	RA0	xx0x xxxx	44, 81

图注： x = 未知， u = 不变， — = 未用， q = 取值视条件而定

注 1： 只有当 BOREN1:BOREN0 配置位 = 01 时， SBOREN 位才可用；否则它将被禁止并读为 0。请参见**第 4.4 节 “欠压复位 （BOR）”**。

2： 只有在特定振荡器配置中才可使用 PLLEN 位，否则，它被禁止并读为 0。请参见**第 2.6.4 节 “INTOSC 模式下的 PLL”**。

3： 只有当禁止主清零复位 （MCLRE 配置位 = 0）时 RA5 位才可用；否则， RA5 读为 0。该位是只读位。

4： 根据不同的主振荡器模式，可单独将 RA6/RA7 及其相关的锁存和方向位配置为端口引脚。当禁止时，这些位读为 0。

5： 通过用户软件或上电复位清零 bit 7 和 bit 6。

6： PWMEN 位的复位状态取决于 CONFIG3L 中 PWMPIN 配置位的设置。

7： 当 WDTEN 配置位使能时该位不起作用。

5.3.5 STATUS 寄存器

如寄存器 5-2 所示，STATUS 寄存器包含 ALU 的算术运算状态。和其他 SFR 一样，它可以是任何指令的操作数。

如果一条影响 Z、DC、C、OV 或 N 位的指令以 STATUS 寄存器作为目标寄存器，指令执行的结果将不会被直接写入，而是根据指令的执行情况来更新 STATUS 寄存器。所以，当执行一条把 STATUS 寄存器作为目标寄存器的指令后，结果可能和预想的不一样。例如， `CLRF STATUS` 将 Z 位置 1 而保持其他状态位不改变 （`000u u1uu`）。

因此，建议仅使用 `BCF`、`BSF`、`SWAPF`、`MOVFF` 和 `MOVWF` 指令来改变 STATUS 寄存器，因为这些指令不会影响 STATUS 寄存器中的 Z、 C、 DC、 OV 或 N 位。

欲知其他不会影响状态位的指令，请参见表 21-2 和表 21-3 中的指令集综述。

注：	在减法运算中，C 和 DC 位分别作为 $\overline{\text{借位}}$ 和辅助借位标志位。

寄存器 5-2： STATUS 寄存器

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(2)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-5 未用：读为 `0`

bit 4 **N**：负标志位
此位用于有符号的算术运算 （以二进制补码方式进行）。它表明结果是否为负 （ALU MSB = `1`）。
`1` = 结果为负
`0` = 结果为正

bit 3 **OV**：溢出标志位
此位用于有符号的算术运算 （以二进制补码方式进行）。表明溢出了 7 位二进制数的范围，溢出将导致符号位 （结果的 bit 7）发生改变。
`1` = 有符号的算术运算发生溢出 （本次运算）
`0` = 没有发生溢出

bit 2 **Z**：全零标志位
`1` = 算术运算或逻辑运算结果为零
`0` = 算术运算或逻辑运算结果不为零

bit 1 **DC**：辅助进位 / $\overline{\text{借位}}$ 标志位 (1)
用于 `ADDWF`、 `ADDLW`、 `SUBLW` 和 `SUBWF` 指令：
`1` = 结果的第 4 个低位发生了进位
`0` = 结果的第 4 个低位未发生进位

bit 0 **C**：进位 / $\overline{\text{借位}}$ 标志位 (2)
用于 `ADDWF`、 `ADDLW`、 `SUBLW` 和 `SUBWF` 指令：
`1` = 结果的最高有效位发生了进位
`0` = 结果的最高有效位未发生进位

注 1： 对于 $\overline{\text{借位}}$，极性是相反的。减法是通过加上第二个操作数的 2 进制补码来实现的。对于移位指令 （`RRF` 和 `RLF`），此位的值来自源寄存器的 bit 4 或 bit 3。

2： 对于 $\overline{\text{借位}}$，极性是相反的。减法是通过加上第二个操作数的 2 进制补码来实现的。对于移位指令 （`RRF` 和 `RLF`），此位的值来自源寄存器的最高位或最低位。

5.4 数据寻址模式

> **注：** 当使能 PIC18 扩展指令集时， PIC18 内核指令集中某些指令的执行会发生改变。更多信息，请参见**第 5.5 节“数据存储器和扩展指令集”**。

数据存储器具有多种寻址模式。大多数指令的寻址模式都是固定的。其他指令最多可以使用 3 种模式，取决于使用哪些操作数以及是否使能了扩展指令集。

寻址模式有：

- 固有寻址
- 立即数寻址
- 直接寻址
- 间接寻址

当使能扩展指令集（XINST 配置位 = 1）时，还可以使用另一种寻址模式，即立即数变址寻址模式。**第 5.5.1 节“用立即数偏移量进行变址寻址”**将更详细地讨论它的操作。

5.4.1 固有和立即数寻址

很多 PIC18 控制指令根本不需要任何参数，执行这些指令要么对整个器件造成影响，要么仅针对一个寄存器进行操作。这种寻址模式就是固有寻址。例如 SLEEP、RESET 和 DAW。

其他指令的工作方式与此类似但需要操作码中有直接参数。由于需要一些立即数作为参数，这种寻址模式被称为立即数寻址。例如 ADDLW 和 MOVLW，它们分别将立即数加到或移入 W 寄存器中。其他的立即数寻址指令，例如 CALL 和 GOTO，包括一个 20 位的程序存储器地址。

5.4.2 直接寻址

直接寻址在操作码中指定操作的全部或部分源地址和 / 或目标地址。此选项由指令附带的参数指定。

在 PIC18 内核指令集中，面向位和字节的指令默认情况下使用直接寻址模式。所有这些指令都包含某个 8 位的直接地址作为它们的最低有效字节。此地址指定数据 RAM 的某个存储区中的寄存器地址（**第 5.3.3 节“通用寄存器”**）或快速操作存储区（**第 5.3.2 节“快速操作存储区”**）中作为指令的数据源的单元地址。

快速操作 RAM 位“a”决定地址的解析方式。当“a”为 1 时，BSR（**第 5.3.1 节“存储区选择寄存器（BSR）”**）的内容和地址一起用于确定寄存器完整的 12 位地址。当“a”为 0 时，此地址将被解释为快速操作存储区中的一个寄存器。使用快速操作 RAM 寻址有时候也被称为直接强制寻址模式。

有几个指令，比如 MOVFF，在操作码中包含完整的 12 位地址（源或目标地址）。在这些情况下，完全忽略 BSR。

操作结果的目标寄存器由目标位“d”确定。当“d”为 1 时，结果被存储到源寄存器并覆盖它原来的内容。当“d”为 0 时，结果被存储在 W 寄存器中。没有“d”参数的指令的目标地址是隐含的，它们是操作针对的寄存器或 W 寄存器。

5.4.3 间接寻址

间接寻址允许用户访问数据存储器中的单元而不需要在指令中给出一个固定的地址。这是通过使用文件选择寄存器（File Select Register， FSR）指向被读取或写入的单元实现的。由于 FSR 本身作为特殊功能寄存器位于 RAM 中，所以也可以在程序控制下直接对它们进行操作。这使得 FSR 对于在数据存储器中实现诸如表和数组等数据结构非常有用。

也可以使用间接指针操作数（ Indirect File Operand，INDF）进行间接寻址。这种操作允许自动递增、递减或偏移指针，从而自动控制指针的值。它通过循环提高代码执行效率，如例 5-5 所示的清零整个 RAM 存储区。

例 5-5： 使用间接寻址将 RAM（BANK 0）清零的方法

```
        LFSR    FSR0, 00h   ;
NEXT    CLRF    POSTINC0    ; Clear INDF
                            ; register then
                            ; inc pointer
        BTFSS   FSR0H, 0    ; All done with
                            ; Bank0?
        BRA     NEXT        ; NO, clear next
CONTINUE                    ; YES, continue
```

5.4.3.1 FSR 寄存器和 INDF 操作数

间接寻址的核心是三组寄存器：FSR0、FSR1和FSR2。每组寄存器都含有一对 8 位寄存器，FSRnH 和 FSRnL。FSRnH 寄存器的高四位未使用，所以每对 FSR 只保存一个 12 位二进制数，从而可以线性寻址整个数据存储空间。因此，FSR 寄存器对被用作数据存储器的地址指针。

间接寻址是通过一组间接指针操作数 （从 INDF0 到 INDF2）完成的。这些操作数可以被看作 “虚拟”寄存器 ：它们是被映射到 SFR 空间中而不是通过物理方式实现的。对特定的 INDF 寄存器执行读或写操作实际上访问的是相应的 FSR 寄存器对。例如，读 INDF1 就是读FSR1H:FSR1L指向的地址单元中的数据。使用INDF寄存器作为操作数的指令实际上使用的是相应的FSR的内容，该内容为指向目标地址的指针。 INDF 操作数只是使用指针的一种较简便的方法。

由于间接寻址使用完整的 12 位地址，因此没有必要进行数据 RAM 分区。因此 BSR 和快速操作 RAM 位对于确定目标地址没有影响。

5.4.3.2 FSR 寄存器和 POSTINC、POSTDEC、 PREINC 以及 PLUSW

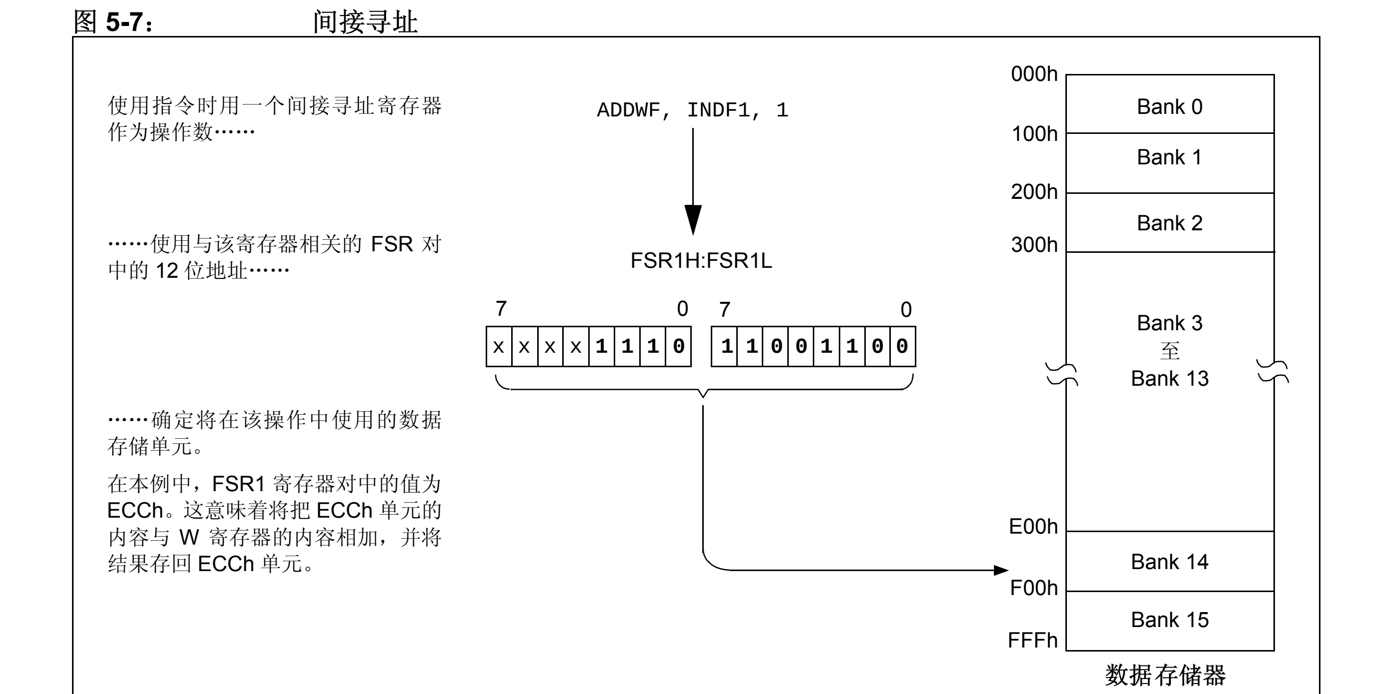
除了 INDF 操作数之外，每对 FSR 寄存器还有四个额外的间接操作数。和 INDF 一样，它们也是不能直接读写的 “虚拟”寄存器。访问这些寄存器其实就是访问对应的FSR寄存器对，并在其所指向的地址单元上进行特定的操作。这些寄存器是：

- POSTDEC：访问 FSR 值，然后自动将它减 1
- POSTINC：访问 FSR 值，然后自动将它加 1
- PREINC：将 FSR 的值加 1，然后在操作中使用该值
- PLUSW：将 W 寄存器中有符号的值 （从 -127 到 128）与 FSR 寄存器中有符号的值相加，并在操作中使用得到的新值。

在本文中访问 INDF 寄存器使用 FSR 寄存器中的值，但不会更改此值。同样，访问 PLUSW 寄存器是将 W 寄存器中的值作为FSR值的偏移量，该操作不会改变这两个寄存器中的值。访问其他虚拟寄存器会更改FSR寄存器的值。

用 POSTDEC、 POSTINC 和 PREINC 对 FSR 进行操作会影响整对寄存器，也就是当 FSRnL 寄存器从 FFh 到 00h 溢出时会向 FSRnH 寄存器进位。但这些操作的结果不会更改 STATUS 寄存器中的标志位（如 Z、N 和 OV 等）。

图 5-7： 间接寻址



PLUSW 寄存器可以用于在数据存储空间实现变址寻址。通过控制 W 寄存器中的值，用户可以访问相对当前指针地址有固定偏移量的地址单元。在某些应用中，该功能可以被用于在数据存储器内部实现某些非常有用的程序控制结构，如软件堆栈。

5.4.3.3 通过 FSR 对其他 FSR 进行操作

在某些特殊情况下，间接寻址操作以其他 FSR 或虚拟寄存器作为寻址目标。例如，使用 FSR 指向一个虚拟寄存器会导致操作不成功。假设如下特殊情况：FSR0H:FSR0L 保存的是 INDF1 的地址 FE7h。尝试使用 INDF0 作为操作数读取 INDF1 的值，将返回 00h。尝试使用 INDF0 作为操作数写入 INDF1，将会导致执行一条 NOP。

另一方面，使用虚拟寄存器对一对 FSR 寄存器进行写操作可能会产生与预期不同的结果。在这些情形下，会将值写入一对 FSR 寄存器，但 FSR 中的值不会有任何递增或递减。因此，写入 INDF2 或 POSTDEC2 时会把同样的值写入 FSR2H:FSR2L。

由于 FSR 是在 SFR 空间中映射的物理寄存器，所以可以通过直接寻址对它们进行操作。用户在使用这些寄存器时应该特别小心，尤其是代码使用间接寻址的情况。

同样，通常允许通过间接寻址对所有其他的 SFR 进行操作。用户在进行此类操作时应该特别小心，以免更改设置从而影响器件操作。

5.5 数据存储器和扩展指令集

使能 PIC18 扩展的指令集（XINST 配置位 = 1）显著改变了数据存储器及其寻址的方式。特别是由于引入了新的数据存储器寻址模式，许多涉及快速操作存储区的 PIC18 内核指令会有所不同。

了解哪些部分保持不变也很重要。数据存储器空间的大小及其线性寻址方式都不会改变。SFR 映射也保持不变。PIC18 内核指令也仍然以直接和间接寻址模式进行操作；固有和立即数指令操作照旧。FSR0 和 FSR1 的间接寻址方式也保持不变。

5.5.1 用立即数偏移量进行变址寻址

使能 PIC18 扩展指令集将更改使用 FSR2 寄存器对在快速操作 RAM 内进行间接寻址的方式。在适当的条件下，使用快速操作存储区的指令（即面向位和字节的指令）可以利用指令中的偏移量来执行变址寻址。这种特定的寻址模式被称为使用立即数偏移量的变址寻址或立即数变址寻址模式。

当使用扩展的指令集时，该寻址模式有如下要求：

- 强制使用快速操作存储区（“a” = 0）；且
- 指针地址参数要小于或等于 5Fh。

在这些条件下，指令的指针地址不被解析为地址的低字节（在直接寻址中和 BSR 一起使用）或快速操作存储区中的 8 位地址。相反，该值被解析为由 FSR2 指定的地址指针的偏移量。该偏移量与 FSR2 的内容相加以获得操作的目标地址。

5.5.2 受到立即数变址寻址模式影响的指令

任何使用直接寻址的 PIC18 内核指令均会受到立即数变址寻址模式的潜在影响。包括所有面向字节和面向位的指令，或标准 PIC18 指令集中几乎一半的指令。只能使用固有或立即数寻址模式的指令不受影响。

此外，如果面向字节和面向位的指令不使用快速操作存储区（快速操作 RAM 位为 1）或包含 60h 以上的地址，它们也不受影响。符合这些条件的指令会像以前一样执行。图 5-8 给出了当使能了扩展的指令集时，各种寻址模式之间的对比。

那些想要在立即数变址寻址模式中使用面向字节或位的指令的用户，应该注意此模式下汇编语法的改变。**第 21.2.1 节“扩展指令的语法”**中对此进行了更详细地说明。

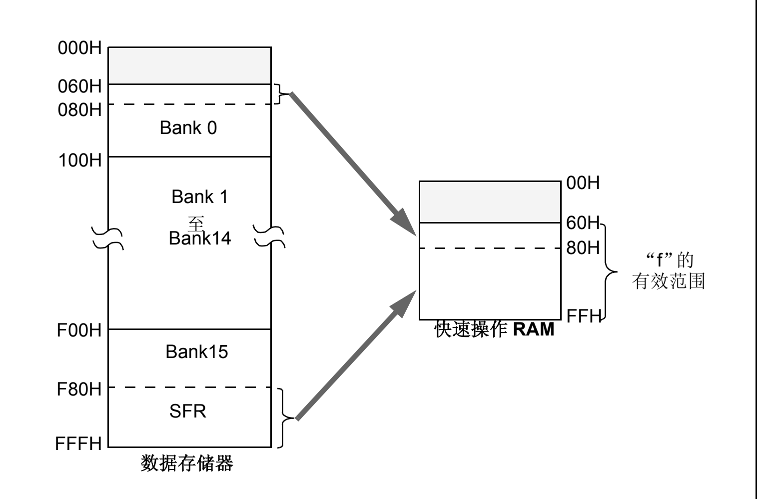
图 5-8： 面向位和字节指令的寻址方式对比 （使能了扩展的指令集）

示例指令：ADDWF, f, d, a（操作码： 0010 01da ffff ffff）

当 a = 0 且 f ≥ 60h 时：

此指令以直接强制模式执行。“f”被解析为快速操作RAM中060h 和 0FFh 之间的单元地址。这实际上是从 060h 到 07Fh （Bank 0）和从 F80h 到 FFFh （Bank 15）的数据存储单元。

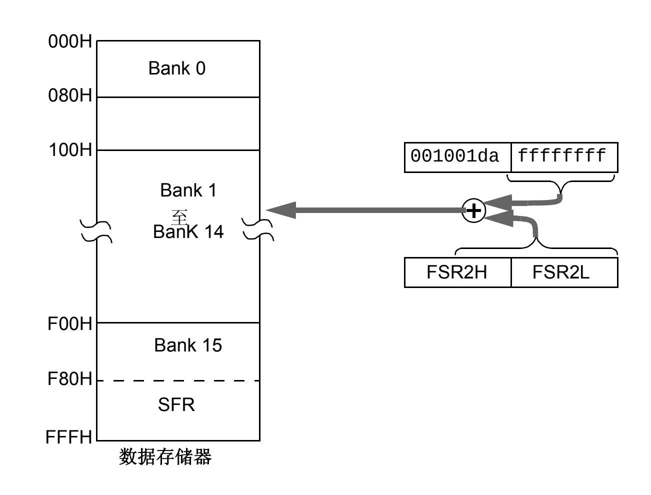
不可用此模式寻址地址低于 60h 的单元。



当 a = 0 且 f ≤ 5Fh 时：

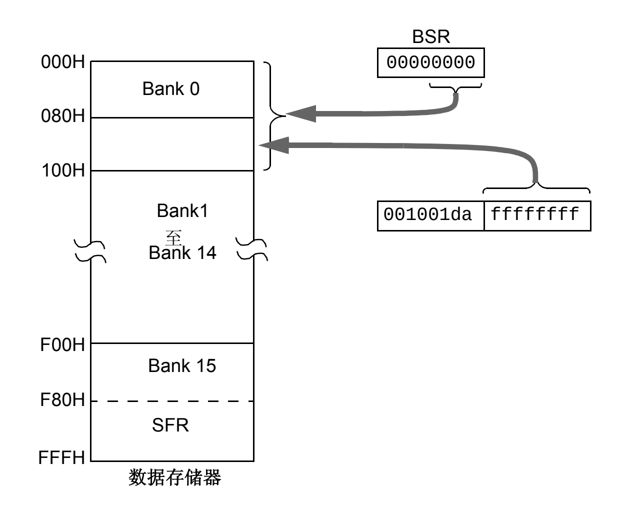
该指令以立即数变址寻址模式执行。“f”被解析为 FSR2 中地址值的偏移量。这两个值相加以获得指令的目标寄存器的地址。此地址可以在数据存储空间的任何地方。

注意在此模式中，正确的语法是：
ADDWF [k], d
其中 “k”就是 “f”。



当 a = 1（f 为任意值）：

指令以直接模式执行 （也被称为直接长地址寻址模式）。“f”被解析为数据存储空间的16个存储区中的一个单元地址。存储区由存储区选择寄存器（BSR）指定。此地址可以在数据存储空间的任何位置。



 超前信息

5.5.3 在立即数变址寻址模式下映射快速操作存储区

使用立即数变址寻址模式能改变快速操作 RAM 前 96 个单元地址（00h 到 5Fh）的映射方式。此模式映射 Bank 0 的内容和由用户定义的可以位于数据存储空间中任何地方的 “窗口” 内容，而不是仅仅包含 Bank 0 底部的内容。FSR2 的值定义映射到窗口的地址的下边界，而上边界则由 FSR2 加 95 （5Fh）决定。地址为 5Fh 以上的快速操作 RAM 的映射方法如前所述 （见第 **5.3.2 节 “快速操作存储区”**）。图 5-9 显示了在此寻址模式中重新映射的快速操作存储区示例。

快速操作存储区的重新映射*仅适用于*立即数变址寻址模式。使用 BSR（快速操作 RAM 位为 `1`）的操作和前面一样继续使用直接寻址模式。

5.6 PIC18 指令执行和扩展的指令集

使能扩展的指令集会在现有的 PIC18 指令集中添加 8 条额外的指令。第 **21.2 节 “扩展的指令集”** 说明了这些指令的执行过程。

图 5-9： 使用立即数变址寻址重新映射快速操作存储区

示例：

```
ADDWF f, d, a
```

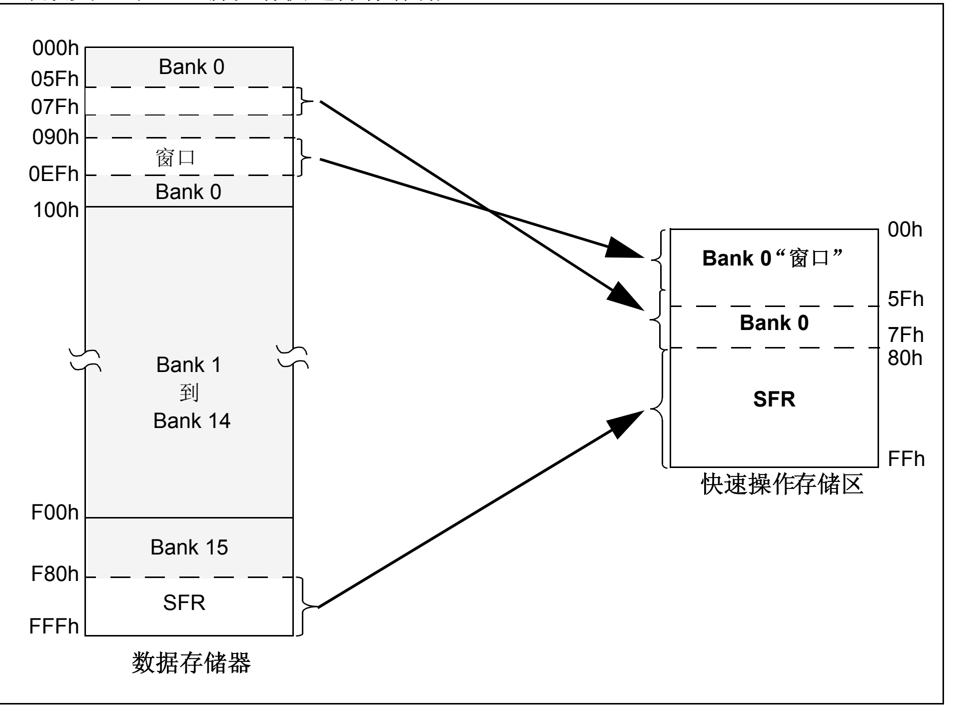
FSR2H:FSR2L = 090h

FSR2 指针指向的地址（090h）到该指针加上 05Fh（0EFh）指向的地址范围内的存储单元被映射到快速操作存储区的低部分地址（000h-05Fh）。

Bank 0 中从 060h 至 07Fh 的单元像往常一样被映射到快速操作存储区的中部。

位于 F80h 至 FFFh 的特殊功能寄存器像往常一样被映射到 80h 到 FFh。

仍然使用 BSR 对地址低于 5Fh 的 Bank 0 中的单元进行寻址。



注：

6.0 闪存程序存储器

在正常工作状态下，闪存程序存储器在整个 VDD 范围内都是可读写并可擦除的。

对程序存储器执行读操作时每次读取一个字节。对程序存储器执行写操作时每次写入一个 8 字节的块。对程序存储器执行擦除操作按照每次 64 字节进行。不允许用户代码执行批量擦除操作。

写或擦除程序存储器将停止取指操作，直到写或擦除操作完成为止。在写或擦除期间不能访问程序存储器，因此无法执行代码。内部编程定时器可终止程序存储器的写入和擦除操作。

写入程序存储器的值不必是有效指令。执行存有无效指令的程序存储器单元会导致执行 NOP 指令。

6.1 表读和表写

为了读写程序存储器，有两种操作可以让处理器在程序存储空间和数据 RAM 之间传送字节。

- 表读 （TBLRD）
- 表写 （TBLWT）

程序存储空间为 16 位宽，而数据 RAM 空间为 8 位宽。表读和表写操作通过一个 8 位寄存器 （TABLAT）在这两个存储空间之间传送数据。

表读操作从程序存储器获取数据并将其存入数据 RAM 空间。图 6-1 给出了在程序存储器和数据 RAM 之间进行表读操作的过程。

表写操作将数据存储空间中的数据存入程序存储器中的保持寄存器。**第 6.5 节"写入闪存程序存储器"**将详细介绍将保持寄存器中的内容写入程序存储器的过程。图 6-2给出了在程序存储器和数据RAM之间进行表写操作的过程。

表操作以字节为单位进行。一个仅包含数据而非程序指令的表块不必字对齐。因此，表块可在任何字节地址处开始和结束。如果使用表写操作将可执行代码写入程序存储器，程序指令就需要进行字对齐。

图 6-1： 表读操作

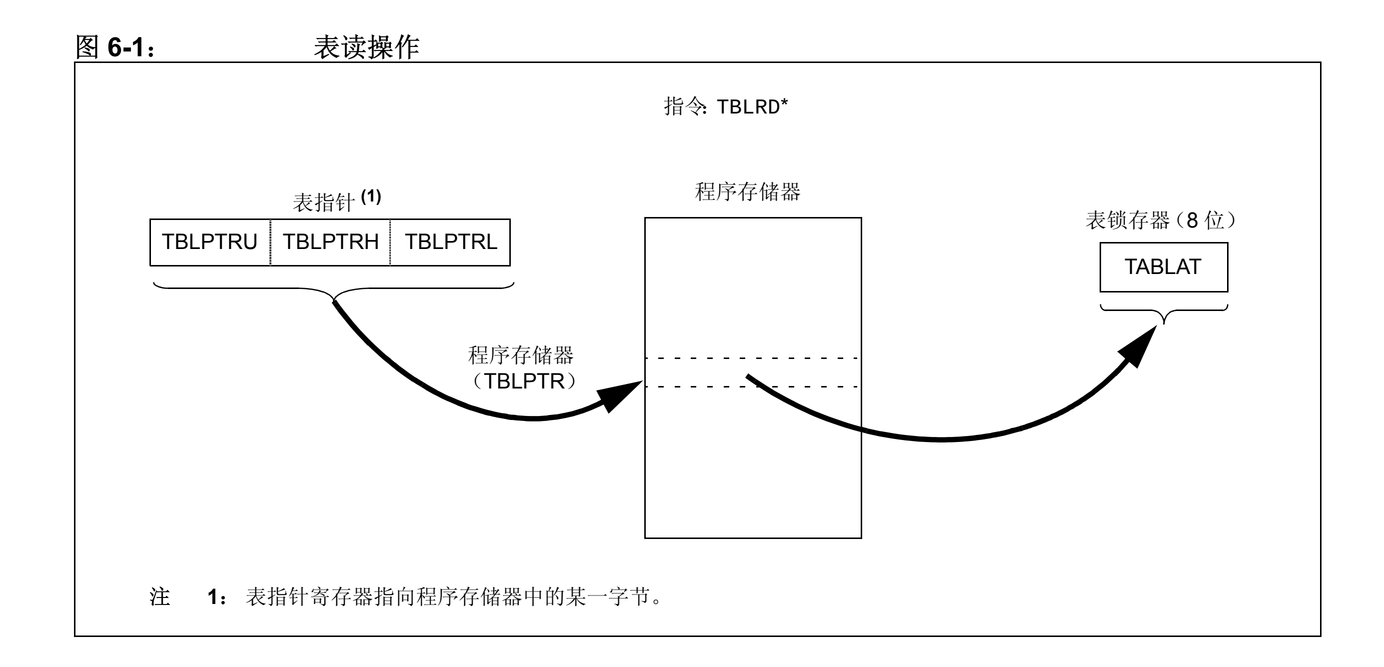
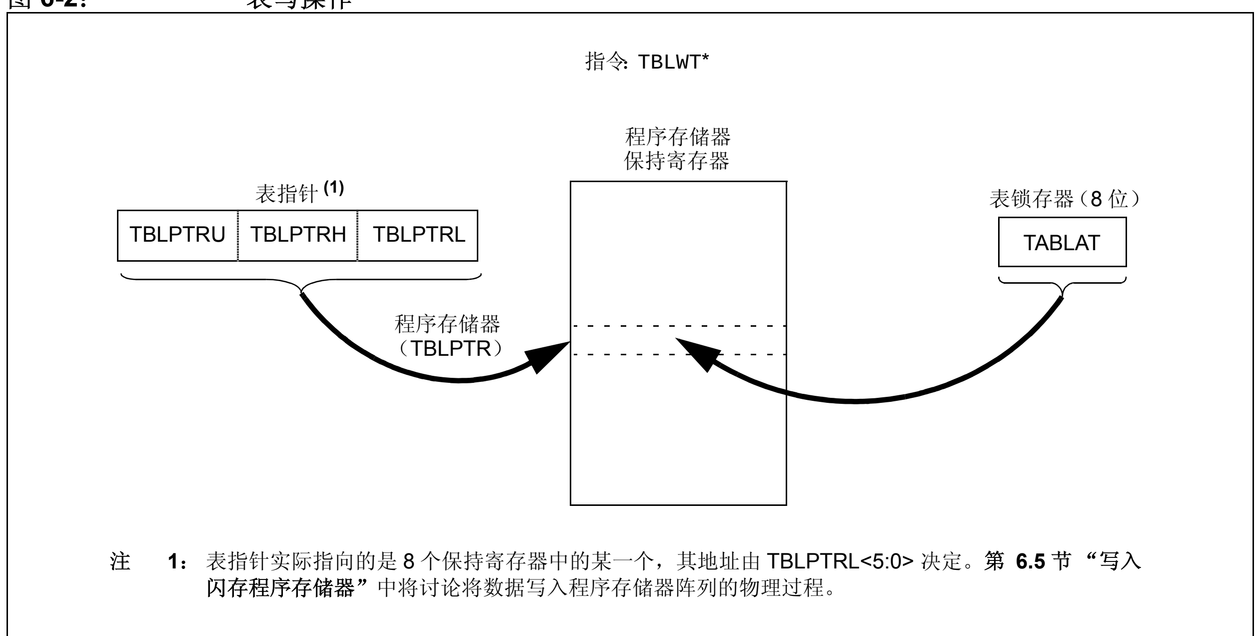


图 6-2： 表写操作



注 1：表指针实际指向的是 8 个保持寄存器中的某一个，其地址由 TBLPTRL<5:0> 决定。第 **6.5** 节 **"写入闪存程序存储器"** 中将讨论将数据写入程序存储器阵列的物理过程。

6.2 控制寄存器

TBLRD 和 TBLWT 指令要用到几个控制寄存器。包括：

- EECON1 寄存器
- EECON2 寄存器
- TABLAT 寄存器
- TBLPTR 寄存器

6.2.1 EECON1 和 EECON2 寄存器

EECON1 寄存器（寄存器 6-1）是访问存储器的控制寄存器。EECON2 寄存器不是物理寄存器，它专用于存储器的擦写过程。读取 EECON2 得到的是全 0。

控制位 EEPGD 决定访问的是程序存储器还是数据 EEPROM 存储器。清零时，所有后续操作针对数据 EEPROM 存储器进行。置 1 时，所有后续操作则针对程序存储器进行。

控制位 CFGS 决定访问的是配置 / 校准寄存器还是程序存储器 / 数据 EEPROM 存储器。置 1 时，后续操作针对配置寄存器进行，与 EEPGD 的值无关（见第 **19.0** 节 **"CPU 的特殊功能"**）。清零时，由 EEPGD 决定将要访问的存储器。

若将FREE位置1，则允许对程序存储器进行擦除操作。擦除操作由下一个 WR 命令触发。当 FREE 位清零时，则只使能写操作。

若将WREN位置1，则允许写操作。上电时将清零WREN位。在 WR 位置 1 时， WRERR 位将被硬件置 1；当内部编程定时器超时并且写操作完成时，清零 WRERR 位。

注：	如果在正常工作期间， WRERR 的读取值为1，则表明写操作因复位而提早终止或进行了非法的写操作。

控制位 WR 用于启动写操作。此位只能由软件置 1 而不能清零。写操作完成后，由硬件将其清零。

注：	当写操作完成时，EEIF中断标志位（PIR2<4>）置 1。此标志位必须由软件清零。

寄存器 6-1： EECON1：EEPROM 控制寄存器 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR[(1)]	WREN	WR	RD
bit 7							bit 0

图注：	S = 可置 1 的位		
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **EEPGD**：闪存程序或数据 EEPROM 存储器选择位
1 = 访问闪存程序存储器
0 = 访问数据 EEPROM 存储器

bit 6 **CFGS**：闪存程序 / 数据 EEPROM 存储器或配置寄存器选择位
1 = 访问配置寄存器
0 = 访问闪存程序存储器或数据 EEPROM 存储器

bit 5 未用：读为 0

bit 4 **FREE**：闪存行擦除使能位
1 = 在下一个 WR 命令时，擦除由 TBLPTR 指定的程序存储器行 （擦除操作完成时清零）
0 = 仅执行写操作

bit 3 **WRERR**：闪存程序 / 数据 EEPROM 错误标志位 [(1)]
1 = 写操作提早终止 （由于正常工作中自定时编程期间的任何复位，或非法写入）
0 = 写操作完成

bit 2 **WREN**：闪存程序 / 数据 EEPROM 写使能位
1 = 允许闪存程序 / 数据 EEPROM 的写周期
0 = 禁止闪存程序 / 数据 EEPROM 的写周期

bit 1 **WR**：写控制位
1 = 启动数据 EEPROM 擦写周期或程序存储器擦写周期。
（该操作是自定时的，一旦写入完成即由硬件将该位清零。软件只能将该位置 1 而不能清零。）
0 = 写入 EEPROM 的周期完成

bit 0 **RD**：读控制位
1 = 启动 EEPROM 读操作。（读取需要一个周期。RD 位由硬件清零。软件只能将 RD 位置 1 而不能清零。当 EEPGD = 1 或 CFGS = 1 时， RD 位无法置 1。）
0 = 不启动 EEPROM 读操作

注 1： 当 WRERR 位置 1 时， EEPGD 和 CFGS 位不会清零，从而允许跟踪错误条件。

6.2.2 TABLAT——表锁存寄存器

表锁存器（Table Latch，TABLAT）是映射到 SFR 空间的一个 8 位寄存器。它用于在程序存储器和数据 RAM 之间传输数据时保存 8 位数据。

6.2.3 TBLPTR——表指针寄存器

表指针 （Table Pointer， TBLPTR）在程序存储器中寻址字节。 TBLPTR 由 3 个 SFR 寄存器组成：表指针最高字节、表指针高字节和表指针低字节（TBLPTRU:TBLPTRH:TBLPTRL）。这 3 个寄存器合起来组成一个 22 位宽的指针。其中低 21 位可使器件寻址至多 2 MB 的程序存储空间。第 22 位则允许访问器件 ID、用户 ID 以及配置位。

TBLRD 和 TBLWT 指令使用表指针寄存器 TBLPTR。利用表操作的四种方法之一，这些指令可以更新 TBLPTR。表 6-1 列出了这些操作。这些操作只会影响 TBLPTR 的低 21 位。

6.2.4 表指针边界

TBLPTR 用于闪存程序存储器的读取、写入和擦除。

当执行 TBLRD 时，表指针的所有 22 位决定将程序存储器哪个单元的数据读入 TABLAT。

当执行 TBLWT 时，表指针寄存器的低 3 位（TBLPTR<2:0>）决定要写入 8 个保持寄存器中的哪一个。当开始定时写入程序存储器时 （通过 WR 位），表指针寄存器的高 19 位 （TBLPTR<21:3>）决定要写入哪一个 8 字节的程序存储块。如需更多详情，请参见**第 6.5 节 "写入闪存程序存储器"**。

当擦除程序存储器时，表指针的高 16 位（TBLPTR<21:6>）指向将要被擦除的 64 字节块。低 6 位（TBLPTR<5:0>）被忽略。

图 6-3 说明了基于闪存程序存储器操作的 TBLPTR 边界。

表 6-1： 使用 TBLRD 和 TBLWT 指令执行表指针操作

示例	表指针操作
TBLRD* TBLWT*	不修改 TBLPTR
TBLRD*+ TBLWT*+	TBLPTR 在读 / 写后递增
TBLRD*- TBLWT*-	TBLPTR 在读 / 写后递减
TBLRD+* TBLWT+*	TBLPTR 在读 / 写前递增

图 6-3： 基于操作的表指针边界



 超前信息

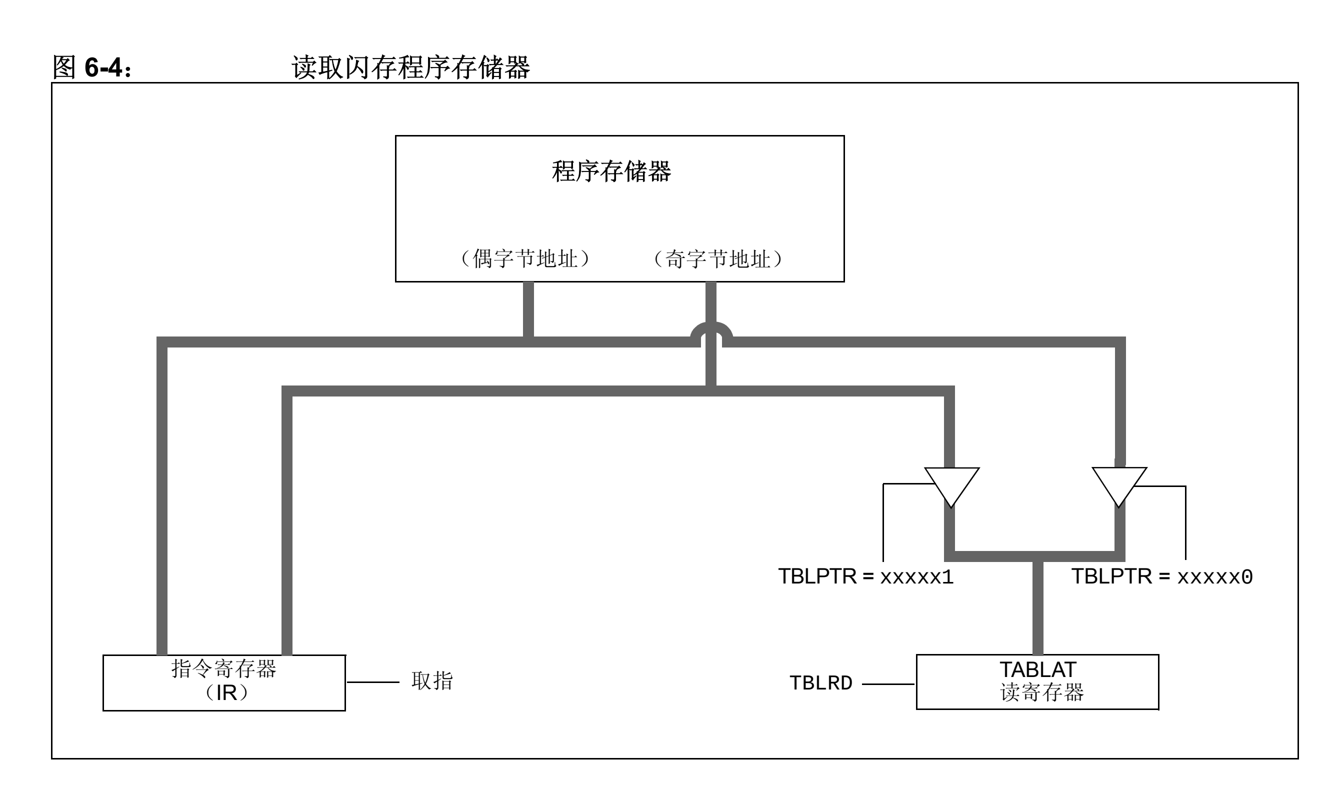
6.3 读取闪存程序存储器

TBLRD 指令用于从程序存储器获取数据并存入数据 RAM。表读操作每次从程序存储器读取一个字节。

TBLPTR 指向程序存储空间内的某个字节。执行 TBLRD 会将指向的字节存入 TABLAT。另外，可以自动修改 TBLPTR 以进行下次表读操作。

内部程序存储器通常是以字为单位构成的。由地址的最低有效位选择字的高字节或者低字节。图 6-4 显示了内部程序存储器和 TABLAT 之间的接口。

图 6-4： 读取闪存程序存储器



例 6-1： 读取一个闪存程序存储器字

```
            MOVLW   CODE_ADDR_UPPER           ; Load TBLPTR with the base
        MOVWF     TBLPTRU                 ; address of the word
        MOVLW     CODE_ADDR_HIGH
        MOVWF     TBLPTRH
        MOVLW     CODE_ADDR_LOW
        MOVWF     TBLPTRL
READ_WORD
        TBLRD*+                           ; read into TABLAT and increment
        MOVF      TABLAT, W               ; get data
        MOVWF     WORD_EVEN
        TBLRD*+                           ; read into TABLAT and increment
        MOVFW     TABLAT, W               ; get data
        MOVF      WORD_ODD
```

6.4 擦除闪存程序存储器

最小擦除块为 32 字（即 64 字节）。只有通过使用外部编程器或通过 ICSP 控制，才能够批量擦除更大的程序存储器块。闪存阵列不支持字擦除。

当从单片机本身启动擦除过程时，将擦除程序存储器的一个64字节的块。TBLPTR<21:6>指向将被擦除的块。TBLPTR<5:0> 被忽略。

擦除操作由 EECON1 寄存器控制。必须将 EEPGD 位置 1 以指向闪存程序存储器。必须将 WREN 位置 1 以使能写操作。必须将 FREE 位置 1 以选择擦除操作。

作为保护机制，必须采用 EECON2 的写操作启动序列。

要擦除内部闪存，需要使用长写周期。在长写周期中，指令执行暂停。内部编程定时器将终止长写周期操作。

6.4.1 闪存程序存储器擦除序列

擦除内部程序存储器块的事件顺序如下：

1. 将要擦除的行地址装入表指针。
2. 设置 EECON1 寄存器以执行擦除操作：
   - 将 EEPGD 位置 1 以指向程序存储器；
   - 将 CFGS 位清零以访问程序存储器；
   - 将 WREN 位置 1 以使能写操作；
   - 将 FREE 位置 1 以使能擦除操作。
3. 禁止中断。
4. 向 EECON2 写入 55h。
5. 向 EECON2 写入 0AAh。
6. 将 WR 位置 1。这将启动行擦除周期。
7. 在擦除操作期间， CPU 将停止工作 （内部定时器计时 2 ms 左右）。
8. 重新允许中断。

例 6-2： 擦除闪存程序存储器行

```
                    MOVLW CODE_ADDR_UPPER          ; load TBLPTR with the base
            MOVWF   TBLPTRU                 ; address of the memory block
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW
            MOVWF   TBLPTRL
ERASE_ROW
            BSF     EECON1, EEPGD           ; point to Flash program memory
            BCF     EECON1, CFGS            ; access Flash program memory
            BSF     EECON1, WREN            ; enable write to memory
            BSF     EECON1, FREE            ; enable Row Erase operation
            BCF     INTCON, GIE             ; disable interrupts
必须的      MOVLW   55h
序列        MOVWF   EECON2                  ; write 55h
            MOVLW   0AAh
            MOVWF   EECON2                  ; write 0AAh
            BSF     EECON1, WR              ; start erase (CPU stall)
            BSF     INTCON, GIE             ; re-enable interrupts
```

6.5 写入闪存程序存储器

最小编程块为 4 字（即 8 字节）。不支持字或字节编程。

表写操作用于将数据装入编程闪存存储器所需的保持寄存器。表写操作使用 8 个保持寄存器进行编程。

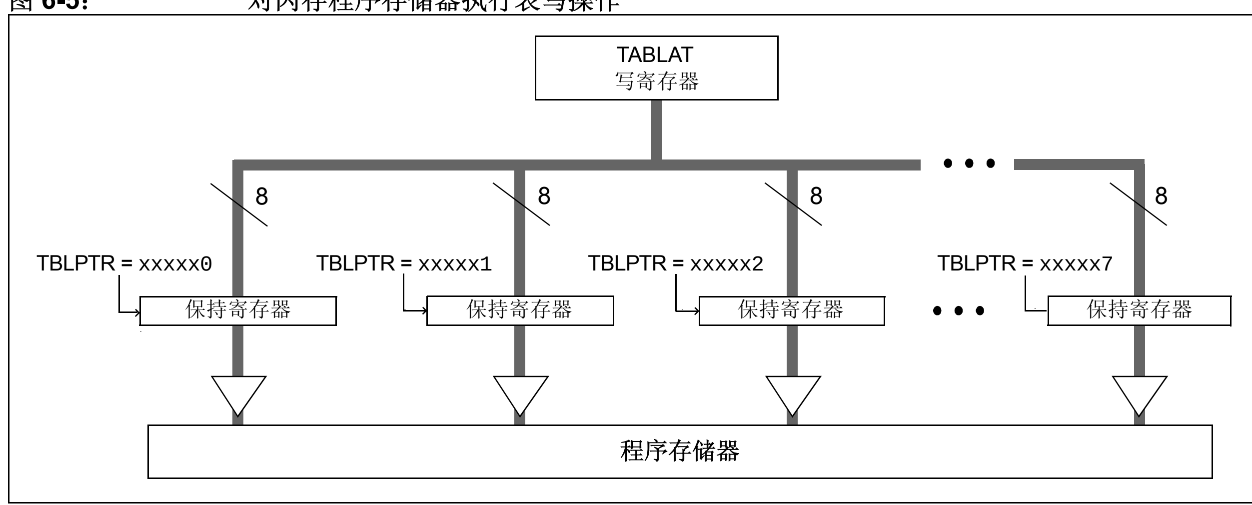
因为表锁存器 （TABLAT）仅一个字节大小，所以每次编程操作必须执行 8 次 TBLWT 指令。由于只写入保持寄存器，所有表写操作实际上都是短周期写操作。在更新完 8 个寄存器后，必须写入 EECON1 寄存器，以便使用长写操作开始编程。

对于内部闪存编程需要使用长写周期。在长写周期中，指令暂停执行。内部编程定时器将终止长写周期操作。

EEPROM 片上定时器控制写操作的时间。写 / 擦除电压由可在器件电压范围内运行的片上电荷泵产生。

> **注：** 器件复位时和写操作完成后，保持寄存器的默认值为 FFh。向保持寄存器写入 FFh 并不会修改寄存器中存储的数值。这意味着，假如不想将某个位从 0 改为 1，仅修改程序存储器中的个别字节也是可以的。当修改个别字节时，在执行写操作前不必重新装入所有的 8 个保持寄存器。

图 6-5：　对闪存程序存储器执行表写操作



6.5.1 闪存程序存储器写操作序列

对内部程序存储器编程的事件序列如下：

1. 将 8 个字节读入 RAM。
2. 必要时更新 RAM 中的数据值。
3. 把要擦除的目标地址装入表指针。
4. 执行行擦除操作。
5. 把要写入的第一个字节的地址装入表指针。
6. 通过自动递增将 8 个字节写入保持寄存器。
7. 设置 EECON1 寄存器以执行写操作：
   - 将 EEPGD 位置 1 以指向程序存储器；
   - 将 CFGS 位清零以访问程序存储器；
   - 将 WREN 位置 1 以使能字节写入。
8. 禁止中断。
9. 向 EECON2 写入 55h。
10. 向 EECON2 写入 0AAh。
11. 将 WR 位置 1，启动写周期。
12. 在写操作期间 CPU 将停止工作 （内部定时器计时 2 ms 左右）。
13. 重新允许中断。
14. 验证存储器 （表读）。

此过程大约需要 6 ms 以更新存储器的一行（8 字节）。例 6-3 给出了所需的代码示例。

> **注：** 在 WR 位置 1 前，表指针需指向保持寄存器中的 8 字节目标地址范围内。

例 6-3： 写入闪存程序存储器

```
                MOVLW   D'8                     ; number of bytes in erase block
                MOVWF   COUNTER
                MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   CODE_ADDR_UPPER         ; Load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF    TABLAT, W               ; get data
                MOVWF   POSTINC0                ; store data
                DECFSZ  COUNTER                 ; done?
                BRA     READ_BLOCK              ; repeat
MODIFY_WORD
                MOVLW   DATA_ADDR_HIGH          ; point to buffer
                MOVWF   FSR0H
                MOVLW   DATA_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   NEW_DATA_LOW            ; update buffer word
                MOVWF   POSTINC0
                MOVLW   NEW_DATA_HIGH
                MOVWF   INDF0
ERASE_BLOCK
                MOVLW   CODE_ADDR_UPPER         ; load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access Flash program memory
                BSF     EECON1, WREN            ; enable write to memory
                BSF     EECON1, FREE            ; enable Row Erase operation
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
必须的          MOVWF   EECON2                  ; write 55h
序列            MOVLW   0AAh
                MOVWF   EECON2                  ; write 0AAh
                BSF     EECON1, WR              ; start erase (CPU stall)
                BSF     INTCON, GIE             ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
                MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
WRITE_BUFFER_BACK
                MOVLW   D’8                     ; number of bytes in holding register
                MOVWF   COUNTER
WRITE_BYTE_TO_HREGS
                MOVFF   POSTINC0, WREG          ; get low byte of buffer data
                MOVWF   TABLAT                  ; present data to table latch
                TBLWT+*                         ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                DECFSZ  COUNTER                 ; loop until buffers are full
                BRA     WRITE_WORD_TO_HREGS
```


例 **6-3**： 写入闪存程序存储器 （续）

```
PROGRAM_MEMORY
              BSF     EECON1, EEPGD          ; point to Flash program memory
              BCF     EECON1, CFGS           ; access Flash program memory
              BSF     EECON1, WREN           ; enable write to memory
              BCF     INTCON, GIE            ; disable interrupts
              MOVLW   55h
   必须的     MOVWF   EECON2                 ; write 55h
   序列       MOVLW   0AAh
              MOVWF   EECON2                 ; write 0AAh
              BSF     EECON1, WR             ; start program (CPU stall)
              BSF     INTCON, GIE            ; re-enable interrupts
              BCF     EECON1, WREN           ; disable write to memory
```

6.5.2 写校验

根据具体的应用，好的编程习惯一般要求使用原始值对写入值进行校验。当连续写入过多的数据已接近规范极限值时，就应该采用写校验。

6.5.3 写操作意外终止

如果由于意外事件 （如掉电或意外复位）终止了写操作，就应该对刚刚编程的存储单元进行验证，如有必要，还要重新进行编程。如果写入操作在正常操作过程中被 MCLR 复位或 WDT 超时复位中断，用户就可以根据需要检测 WRERR 位并重新写入。

6.5.4 避免误写操作

为了防止误写闪存程序存储器，必须遵循写操作启动序列。更多详情，请参见**第 19.0 节“CPU 的特殊功能”**。

6.6 代码保护时的闪存程序存储器操作

如需了解有关闪存程序存储器代码保护的详情，请参见**第 19.5 节 “程序校验和代码保护”**。

表 **6-2**： 与闪存程序存储器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
TBLPTRU	—	—	bit 21	程序存储器表指针最高字节 （TBLPTR<20:16>）					41
TBPLTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								41
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								41
TABLAT	程序存储器表锁存器								41
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
EECON2	EEPROM 控制寄存器 2 （非物理寄存器）								43
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	43
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	43
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	43
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	43

图注： — = 未用位 （读为 0）。访问闪存 /EEPROM 存储器时不使用阴影单元。

注：

 超前信息

7.0 数据 EEPROM 存储器

数据 EEPROM 在整个 VDD 范围内正常运行时是可读写的。该数据存储器并不直接映射到寄存器文件空间。而是通过特殊功能寄存器 （SFR）来间接寻址。

有四个 SFR 用于读写程序和数据 EEPROM 存储器。这些寄存器是：

- EECON1
- EECON2
- EEDATA
- EEADR

EEPROM 数据存储器允许按字节读写。在对数据存储器模块操作时， EEDATA 内存放 8 位读写数据，而 EEADR 寄存器存放要访问的 EEPROM 地址。这些器件具有 128 字节的数据 EEPROM，地址范围从 00h 到 FFh。

EEPROM 数据存储器耐擦写能力很强。字节的写操作将自动擦除相应单元并写入新值（即先擦后写）。写入时间由片上定时器控制，它会随着电压、温度以及芯片的不同而变化。请参见参数 D122 （第 **22.0** 节 **“电气规范”**中的表 22-1 ）以了解具体的数值。

7.1 EEADR 寄存器

EEPROM 地址寄存器可寻址 256 字节的数据 EEPROM。

7.2 EECON1 和 EECON2 寄存器

EECON1 是存储器访问控制寄存器。

EECON2 不是物理寄存器。读取 EECON2 将得到全 0。EECON2 寄存器仅用于存储器的写和擦除过程。

控制位 EEPGD 决定访问的是程序存储器还是数据 EEPROM 存储器。清零时，访问数据 EEPROM 存储器；置 1 时，则访问程序存储器。

控制位 CFGS 决定访问的是配置寄存器还是程序存储器 / 数据 EEPROM 存储器。置 1 时，后续操作会访问配置寄存器。而当 CFGS 清零时，则由 EEPGD 位选择闪存程序存储器或数据 EEPROM 存储器。

擦除和写入操作由 WREN 位使能和禁止。该位置 1 时，允许擦除和写入操作； 清零时，禁止擦除和写入操作。只要 WREN 位清零，WR 位就无法置 1。这种机制有助于防止由于错误 （意外）执行代码而误写存储器。

除了开始擦除或写入操作外，固件必须始终保持 WREN 位清零。一旦固件已经将 WR 位置 1，WREN 位就可以被清零。将 WREN 位清零不会影响进行中的操作。

当写操作由于复位中断时， WRERR 位会置 1。在这种情况下，用户可以检查 WRERR 位，并重写相应的单元。由于复位已使数据寄存器和地址寄存器 （EEDATA 和 EEADR）清零，因此必须对它们进行重载。

控制位 RD 和 WR 分别启动读和擦 / 写操作。这些位由固件置 1，并在操作完成时由硬件清零。

当访问程序存储器（EEPGD = 1）时，RD 位无法置 1。程序存储器是通过表读指令读取的。有关表读操作的信息，请参见第 **6.1** 节 **“表读和表写”**。

注：	写操作完成后，PIR2 寄存器中的中断标志位 EEIF 会置 1。此标志位必须由软件清零。

寄存器 7-1： EECON1：EEPROM 控制寄存器 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR[(1)]	WREN	WR	RD
bit 7							bit 0

图注：	S = 可置 1 的位		
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **EEPGD**：闪存程序或数据 EEPROM 存储器选择位
1 = 访问闪存程序存储器
0 = 访问数据 EEPROM 存储器

bit 6 **CFGS**：闪存程序 / 数据 EEPROM 存储器或配置寄存器选择位
1 = 访问配置寄存器
0 = 访问闪存程序存储器或数据 EEPROM 存储器

bit 5 未用：读为 0

bit 4 **FREE**：闪存行擦除使能位
1 = 在下一个 WR 命令时，擦除由 TBLPTR 指定的程序存储器行 （擦除操作完成时清零）
0 = 仅执行写操作

bit 3 **WRERR**：EEPROM 错误标志位 [(1)]
1 = 写操作提早终止 （自定时擦除或编程操作过程中的 $\overline{\text{MCLR}}$ 或 WDT 复位）
0 = 写操作完成

bit 2 **WREN**：擦 / 写使能位
1 = 允许擦 / 写周期
0 = 禁止擦 / 写周期

bit 1 **WR**：写控制位
1 = 启动数据EEPROM擦写周期或程序存储器擦写周期。（该操作是自定时的，一旦写入完成即由硬件将该位清零。软件只能将该位置 1 而不能清零。）
0 = 写周期完成

bit 0 **RD**：读控制位
1 = 启动 EEPROM 读操作。（读取需要一个周期。RD 位由硬件清零。软件只能将 RD 位置 1 而不能清零。当 EEPGD = 1 时， RD 位无法置 1。）
0 = 读操作完成

注 1： 当 WRERR 位置 1 时， EEPGD 或 FREE 位不会清零，从而允许跟踪错误条件。

7.3 读数据 EEPROM 存储器

要读取数据存储器单元，用户必须将地址写入 EEADR 寄存器，清零 EEPGD 控制位（EECON1<7>）然后将控制位 RD（EECON1<0>）置 1。由于该数据在下一个指令周期才可用；因此，EEDATA 寄存器可由下一条指令读取。EEDATA 将保持这个值直到另一次读操作开始，或直到它被用户写入（在写操作过程中）。

7.4 写数据 EEPROM 存储器

要写 EEPROM 数据单元，必须首先将地址写入 EEADR 寄存器并将数据写入 EEDATA 寄存器。必须遵循例 7-2 中的执行序列才能开始写周期。

如果未完全按照以上序列（将 55h 写入 EECON2，将 0AAh 写入 EECON2，然后将 WR 位置 1）逐字节写入，写操作将不会开始。在执行该代码段时，强烈建议禁止中断。

此外，EECON1 中的 WREN 位也必须被置 1 以使能写操作。这种机制可防止由于意外执行代码（即程序跑飞）造成对数据 EEPROM 的误写入。除了在更新 EEPROM 时，否则 WREN 位应该一直保持清零状态。WREN 位不能由硬件清零。

写序列开始以后，就不能修改 EECON1、EEADR 和 EDATA 了。除非 WREN 位置 1，否则将不允许 WR 位置 1。WREN 位必须在前面的指令中置 1，WR 和 WREN 位不能由同一指令置 1。

写周期完成后，WR 位将由硬件清零，同时 EEPROM 中断标志位（EEIF）被置 1。用户可以允许该中断或对 WR 位进行查询。EEIF 必须由软件清零。

7.5 写校验

根据具体的应用，好的编程习惯一般要求使用原始值对写入值进行校验。当多次写入已接近规范极限值时，就应该采用写入校验。

7.6 避免误写操作

有些情况下，用户可能不希望向数据 EEPROM 存储器写入数据。为了防止误写 EEPROM，器件内建了各种保护机制。上电时，WREN 位被清零。此外，上电延时定时器延时期间（延时 72 ms）也禁止对 EEPROM 执行写操作。

在电源欠压、电源不稳定或软件故障期间，写操作启动序列和 WREN 位可共同防止误写操作的发生。

例 7-1： 读数据 EEPROM

```
        MOVLW   DATA_EE_ADDR              ;
        MOVWF   EEADR                     ; Data Memory Address to read
        BCF     EECON1, EEPGD             ; Point to DATA memory
        BSF     EECON1, RD                ; EEPROM Read
        MOVF    EEDATA, W                 ; W = EEDATA
```

例 7-2： 写数据 EEPROM

```
            MOVLW   DATA_EE_ADDR          ;
            MOVWF   EEADR                 ; Data Memory Address to write
            MOVLW   DATA_EE_DATA          ;
            MOVWF   EEDATA                ; Data Memory Value to write
            BCF     EECON1, EEPGD         ; Point to DATA memory
            BSF     EECON1, WREN          ; Enable writes
            BCF     INTCON, GIE           ; Disable Interrupts
            MOVLW   55h                   ;
必须的      MOVWF   EECON2                ; Write 55h
序列        MOVLW   0AAh                  ;
            MOVWF   EECON2                ; Write 0AAh
            BSF     EECON1, WR            ; Set WR bit to begin write
            BSF     INTCON, GIE           ; Enable Interrupts

            SLEEP                         ; Wait for interrupt to signal write complete
            BCF     EECON1, WREN          ; Disable writes
```

7.7 代码保护时的操作

数据 EEPROM 存储器在配置字中有它自己的代码保护位。如果使能了代码保护，将禁止对其进行外部读写操作。

不管代码保护配置位的状态如何，单片机本身可以读写内部数据 EEPROM。更多信息请参见**第 19.0 节“CPU 的特殊功能”**。

7.8 使用数据 EEPROM

数据 EEPROM 是具有高耐擦写能力、可按字节寻址的阵列，经优化允许频繁地更改存储信息（例如，程序变量或其他经常更新的数据）。更新频率通常要高于规范 D124 或 D124A 的规定。如果情况并非如此，就必须执行阵列刷新。因此，不常修改的变量（例如常数、ID、校准值等）应该存储在闪存程序存储器中。

例 7-3 所示为简单的数据 EEPROM 刷新程序。

注：	如果数据 EEPROM 仅用于存储常数和 / 或很少修改的数据，可能不必进行阵列刷新。请参见 D124 或 D124A 规范。

例 7-3： 数据 EEPROM 刷新程序

```
        CLRF    EEADR                       ; Start at address 0
        BCF     EECON1, CFGS                ; Set for memory
        BCF     EECON1, EEPGD               ; Set for Data EEPROM
        BCF     INTCON, GIE                 ; Disable interrupts
        BSF     EECON1, WREN                ; Enable writes
LOOP                                        ; Loop to refresh array
        BSF     EECON1, RD                  ; Read current address
        MOVLW   55h                         ;
        MOVWF   EECON2                      ; Write 55h
        MOVLW   0AAh                        ;
        MOVWF   EECON2                      ; Write 0AAh
        BSF     EECON1, WR                  ; Set WR bit to begin write
        BTFSC   EECON1, WR                  ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F                    ; Increment address
        BRA     Loop                        ; Not zero, do it again

        BCF     EECON1, WREN                ; Disable writes
        BSF     INTCON, GIE                 ; Enable interrupts
```

表 7-1： 与数据 EEPROM 存储器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
EEADR	EEPROM 地址寄存器								43
EEDATA	EEPROM 数据寄存器								43
EECON2	EEPROM 控制寄存器 2 （非物理寄存器）								43
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	43
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	43
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	43
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	43

图注： x = 未知， u = 不变， — = 未用 （读为 0）。在访问闪存 /EEPROM 时不使用阴影单元。

 超前信息

8.0 8 x 8 硬件乘法器

8.1 简介

所有的 PIC18 器件均包含一个 8 x 8 硬件乘法器（乘法器是 ALU 的一部分）。该乘法器可执行无符号运算并产生一个 16 位运算结果，该结果存储在一对乘积寄存器 PRODH:PRODL 中。该乘法器执行的运算不会影响 STATUS 寄存器中的任何标志位。

通过硬件执行乘法运算只需要 1 个指令周期。硬件乘法器具有更高的计算吞吐量并减少了乘法算法的代码长度，从而可在许多先前仅能使用数字信号处理器的应用中使用 PIC18 器件。表 8-1 给出了各种硬件和软件乘法运算的比较，包括所需的存储空间和执行时间。

8.2 工作原理

例 8-1 给出了一个 8 x 8 无符号乘法运算的指令序列。当已在 WREG 寄存器中装入了一个乘数时，实现该运算仅需一条指令。

例 8-2 给出了执行 8 x 8 有符号乘法运算的指令序列。要弄清乘数的符号位，必须检查每个乘数的最高有效位（MSb），并做相应的减法。

例 8-1： 8 x 8 无符号乘法程序

```
MOVF    ARG1, W     ;
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
```

例 8-2： 8 x 8 有符号乘法程序

```
MOVF    ARG1, W
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
BTFSC   ARG2, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG2
```

表 8-1： 各种乘法运算的性能比较

程序	乘法实现方法	程序存储空间（字数）	周期数（最多）	时间		
				@40 MHz	**@10 MHz**	**@4 MHz**
8 x 8 无符号	软件乘法	13	69	6.9 μs	27.6 μs	69 μs
	硬件乘法	1	1	100 ns	400 ns	1 μs
8 x 8 有符号	软件乘法	33	91	9.1 μs	36.4 μs	91 μs
	硬件乘法	6	6	600 ns	2.4 μs	6 μs
16 x 16 无符号	软件乘法	21	242	24.2 μs	96.8 μs	242 μs
	硬件乘法	28	28	2.8 μs	11.2 μs	28 μs
16 x 16 有符号	软件乘法	52	254	25.4 μs	102.6 μs	254 μs
	硬件乘法	35	40	4.0 μs	16.0 μs	40 μs

例 8-3 给出了 16 x 16 无符号乘法运算的指令序列。公式 8-1 为所使用的算法。32 位运算结果存储在 4 个寄存器（RES3:RES0）中。

公式 8-1： 16 x 16 无符号乘法算法

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

例 8-3： 16 x 16 无符号乘法程序

```
    MOVF    ARG1L, W
    MULWF   ARG2L       ; ARG1L * ARG2L->
                        ; PRODH:PRODL
    MOVFF   PRODH, RES1;
    MOVFF   PRODL, RES0;
;
    MOVF    ARG1H, W
    MULWF   ARG2H       ; ARG1H * ARG2H->
                        ; PRODH:PRODL
    MOVFF   PRODH, RES3;
    MOVFF   PRODL, RES2;
;
    MOVF    ARG1L, W
    MULWF   ARG2H       ; ARG1L * ARG2H->
                        ; PRODH:PRODL
    MOVF    PRODL, W    ;
    ADDWF   RES1, F     ; Add cross
    MOVF    PRODH, W    ; products
    ADDWFC  RES2, F     ;
    CLRF    WREG        ;
    ADDWFC  RES3, F     ;
;
    MOVF    ARG1H, W    ;
    MULWF   ARG2L       ; ARG1H * ARG2L->
                        ; PRODH:PRODL
    MOVF    PRODL, W    ;
    ADDWF   RES1, F     ; Add cross
    MOVF    PRODH, W    ; products
    ADDWFC  RES2, F     ;
    CLRF    WREG        ;
    ADDWFC  RES3, F     ;
```

例 8-4 给出了 16 x 16 有符号乘法运算的指令序列。公式 8-2 为所使用的算法。32 位运算结果存储在 4 个寄存器（RES3:RES0）中。要弄清乘数的符号位，必须检查每个乘数的最高有效位（MSb），并做相应的减法。

公式 8-2： 16 x 16 有符号乘法算法

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H<7> \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H<7> \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

例 8-4： 16 x 16 有符号乘法程序

```
    MOVF    ARG1L, W
    MULWF   ARG2L       ; ARG1L * ARG2L ->
                        ; PRODH:PRODL
    MOVFF   PRODH, RES1;
    MOVFF   PRODL, RES0;
;
    MOVF    ARG1H, W
    MULWF   ARG2H       ; ARG1H * ARG2H ->
                        ; PRODH:PRODL
    MOVFF   PRODH, RES3;
    MOVFF   PRODL, RES2;
;
    MOVF    ARG1L, W
    MULWF   ARG2H       ; ARG1L * ARG2H ->
                        ; PRODH:PRODL
    MOVF    PRODL, W    ;
    ADDWF   RES1, F     ; Add cross
    MOVF    PRODH, W    ; products
    ADDWFC  RES2, F     ;
    CLRF    WREG        ;
    ADDWFC  RES3, F     ;
;
    MOVF    ARG1H, W    ;
    MULWF   ARG2L       ; ARG1H * ARG2L ->
                        ; PRODH:PRODL
    MOVF    PRODL, W    ;
    ADDWF   RES1, F     ; Add cross
    MOVF    PRODH, W    ; products
    ADDWFC  RES2, F     ;
    CLRF    WREG        ;
    ADDWFC  RES3, F     ;
;
    BTFSS   ARG2H, 7    ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1   ; no, check ARG1
    MOVF    ARG1L, W    ;
    SUBWF   RES2        ;
    MOVF    ARG1H, W    ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7    ; ARG1H:ARG1L neg?
    BRA     CONT_CODE   ; no, done
    MOVF    ARG2L, W    ;
    SUBWF   RES2        ;
    MOVF    ARG2H, W    ;
    SUBWFB  RES3
;
CONT_CODE
   :
```


9.0 I/O 端口

根据选定的器件和使能的功能的不同，最多有 5 个端口可供使用。I/O 端口的某些引脚与器件的外设功能复用。通常，当外设使能时，对应的引脚就无法用作通用 I/O 引脚。

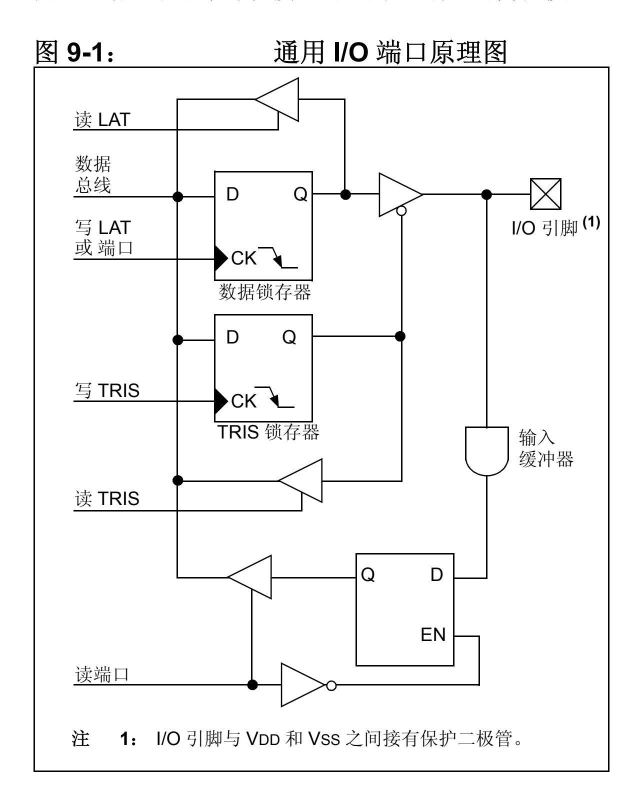
每个端口均有 3 个用于控制其操作的寄存器。它们是：

- TRIS 寄存器 （数据方向寄存器）
- PORT 寄存器 （读取器件引脚的电平）
- LAT 寄存器 （输出锁存器）

在对 I/O 引脚的驱动值进行读 - 修改 - 写操作时会用到数据锁存器 （LAT 寄存器）。

图 9-1给出了不带外设接口的通用I/O端口的简化模型。

图 9-1： 通用 I/O 端口原理图



9.1 PORTA、 TRISA 和 LATA 寄存器

PORTC 是 8 位宽的双向端口。相应的数据方向寄存器是 TRISA。将 TRISA 置 1（= 1）可以把对应的 PORTA 引脚设置为输入引脚（即，将对应的输出驱动器置于高阻态）。将 TRISA 位清零 （= 0）将使对应的 PORTA 引脚作为输出引脚（即，将输出锁存器的内容从所选择的引脚输出）。

读 PORTA 寄存器读入的是引脚的状态，而写该寄存器是将数据写入端口锁存器。

数据锁存器（LATA）也是存储器映射的。对 LATA 寄存器执行读 - 修改 - 写操作将读写 PORTA 的输出锁存值。

引脚 RA6 和 RA7 与主振荡器引脚复用；通过在配置寄存器中选择主振荡器可以将这两个引脚使能为振荡器引脚或 I/O 引脚（详情见第 **19.1** 节 **"配置位"**）。当不被用作端口引脚时，RA6 和 RA7 及其对应的 TRIS 和 LAT 位读为 0。

RA0 引脚与一个模拟输入、外部中断输入、电平变化中断输入以及模拟比较器输入复用，成为 RA0/AN0/INT0/KBI0/CMP0 引脚。

RA1 引脚与一个模拟输入、外部中断输入和电平变化中断输入复用，成为 RA1/AN1/INT1/KBI1 引脚。

引脚 RA2 和 RA3 与增强型 USART 发送和接收输入复用 （详情见第 **19.1** 节 **"配置位"**）。

RA4 引脚与 Timer0 模块时钟输入、一个模拟输入和模拟VREF+ 输入复用，成为RA4/T0CKI/AN2/VREF+引脚。

PWM 故障检测输入引脚 $\overline{\text{FLTA}}$ 与 RA5 和 RA7 引脚复用。通过将配置寄存器 3H 中的 FLTAMX 位清零和置 1 可决定该引脚的配置。

注：	在上电复位时，RA0、RA1、RA4 和 RA5 被配置为模拟输入并读为 0。 RA2 和 RA3 被配置为数字输入。

即使在 PORTA 引脚被用作模拟输入的时候，TRISA 寄存器仍然控制 PORTA 引脚的方向。在将它们用作模拟输入时，用户必须确保 TRISA 寄存器中的位保持为置 1 状态。

例 9-1： 初始化 PORTA

```
CLRF   PORTA   ; Initialize PORTA by
               ; clearing output
               ; data latches
CLRF   LATA    ; Alternate method
               ; to clear output
               ; data latches
MOVLW  07h     ; Configure A/D
MOVWF  ADCON1  ; for digital inputs
MOVWF  07h     ; Configure comparators
MOVWF  CMCON   ; for digital input
MOVLW  0CFh    ; Value used to
               ; initialize data
               ; direction
MOVWF  TRISA   ; Set RA<7:6,3:0> as inputs
               ; RA<5:4> as outputs
```

表 9-1： PORTA I/O 汇总

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RA0/AN0/INT0/KBI0/CMP0	RA0	0	O	DIG	LATA<0> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<0> 数据输入；使能模拟输入时被禁止。
	AN0	1	I	ANA	模拟输入 0。
	INT0	1	I	ST	外部中断 0。
	KBI0	1	I	TTL	电平变化中断引脚。
	CMP0	1	I	ANA	比较器 0 输入。
RA1/AN1/INT1/KBI1	RA1	0	O	DIG	LATA<1> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<1> 数据输入；使能模拟输入时被禁止。
	AN1	1	I	ANA	模拟输入 1。
	INT1	1	I	ST	外部中断 1。
	KBI1	1	I	TTL	电平变化中断引脚。
RA2/TX/CK	RA2	0	O	DIG	LATA<2> 数据输出；不受模拟输入影响。当使能 CVREF 输出时禁止。
		1	I	TTL	PORTA<2> 数据输入。当使能模拟功能或使能 CVREF 输出时被禁止。
	TX	0	0	DIG	EUSART 异步发送。
	CK	0	O	DIG	EUSART 同步时钟。
		1	I	ST	
RA3/RX/DT	RA3	0	O	DIG	LATA<3> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<3> 数据输入；使能模拟输入时被禁止。
	RX	1	I	ANA	EUSART 异步接收。
	DT	0	O	DIG	EUSART 同步数据。
		1	I	TTL	
RA4/T0CKI/AN2/VREF+	RA4	0	O	DIG	LATA<4> 数据输出。
		1	I	ST	PORTA<4> 数据输入； POR 时为默认配置。
	T0CKI	1	I	ST	Timer0 外部时钟输入。
	AN2	1	I	ANA	模拟输入 2。
	VREF+	1	I	ANA	A/D 参考电压 （高电平端）输入。
$\overline{\text{MCLR}}$/VPP/RA5/FLTA	$\overline{\text{MCLR}}$	1	I	ST	主清零 （复位）输入。此引脚为低电平时器件复位。
	VPP	1	I	ANA	编程电压输入。
	RA5	1	I	ST	数字输入。
	$\overline{\text{FLTA}}$[1]	1	I	ST	PWM 故障检测输入。
RA6/OSC2/CLKO/T1OSO/T1CKI/AN3	RA6	0	O	DIG	LATA<6> 数据输出。仅在 RCIO、 INTIO2 和 ECIO 模式下使能。
		1	I	ST	PORTA<6> 数据输入。仅在 RCIO、 INTIO2 和 ECIO 模式下使能。
	OSC2	0	O	ANA	晶振输出或外部时钟源输出。
	CLKO	0	O	ANA	振荡器晶振输出。
	T1OSO[2]	0	O	ANA	Timer1 振荡器输出。
	T1CKI[2]	1	I	ST	Timer1 时钟输入。
	AN3	1	I	ANA	模拟输入 3。
RA7/OSC1/CLKI/T1OSI/FLTA	RA7	0	O	DIG	LATA<7> 数据输出。外部振荡器模式下禁止。
		1	I	TTL	PORTA<7> 数据输入。外部振荡器模式下禁止。
	OSC1	1	I	ANA	振荡器晶振输入或外部时钟源输入。
	CLKI	1	I	ANA	外部时钟源输入。
	T1OSI[2]	1	I	ANA	Timer1 振荡器输入。
	$\overline{\text{FLTA}}$[1]	1	I	ST	PWM 故障检测输入。

图注： DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发器输入缓冲器； ANA = 模拟电平输入 / 输出；
x = 无关位 （TRIS 位不影响端口方向或者在此可忽略）。

注 1： $\overline{\text{FLTA}}$ 的设置取决于 CONFIG3H 中 FLTAMX 配置位的值。

2： T1OSI 和 T1OSO/T1CKI 的设置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

表 **9-2**： 与 **PORTA** 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
PORTA	RA7[(1)]	RA6[(1)]	RA5	RA4	RA3	RA2	RA1	RA0	44
LATA	LATA7[(1)]	LATA6[(1)]	PORTA 数据锁存器 （读取和写入数据锁存器）						43
TRISA	TRISA7[(1)]	TRISA6[(1)]	PORTA 数据方向控制寄存器						43
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	41
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	42
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	42
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	42

图注： — = 未用 （读为 0）。 PORTA 未使用阴影单元。

注 **1**： RA7:RA6 及其相关的锁存器和数据方向位根据振荡器配置使能为 I/O 引脚；否则，它们读为 0。

9.2 PORTB、 TRISB 和 LATB 寄存器

PORTB 是 8 位宽的双向端口。对应的数据方向寄存器是TRISB。将TRISB位置1（= 1）可以使对应的PORTB 引脚作为输入引脚（即，将对应的输出驱动器置于高阻态）。将 TRISB 位清零 （= 0）将使对应的 PORTB 引脚作为输出引脚（即，将输出锁存器的内容从所选择的引脚输出）。

数据锁存器 （LATB）也是存储器映射的。对 LATB 寄存器的读－修改－写操作将读写 PORTB 的输出锁存值。

例 9-2： 初始化 PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Fh     ; Set RB<4:0> as
MOVWF   ADCON1  ; digital I/O pins
                ; (required if config bit
                ; PBADEN is set)
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

PORTB 的每个引脚都有内部弱上拉电路。单个控制位可以开启所有上拉电路。可以通过清零 RBPU 位（INTCON2<7>）来开启上拉电路。当端口引脚被配置为输出时，其弱上拉电路会自动切断。此弱上拉功能在上电复位时被禁止。

注：	在上电复位时，除 RB2 和 RB3 外，PORTB 被配置为数字输入。 当配置寄存器 3H 中的 T1OSCMX 位清零时， RB2 和 RB3 被配置为模拟输入。否则， RB2 和 RB3 也被配置为数字输入。

引脚RB0、RB1和RB4:RB7与电源控制PWM输出复用。

引脚 RB2 和 RB3 与外部中断输入、电平变化中断输入、模拟比较器输入以及 Timer1 振荡器输入和输出复用，分别成为 RB2/INT2/KBI2/CMP2/T1OSO/T1CKI 和 RB3/INT3/KNBI3/CMP1/T1OSI。

当电平变化中断功能使能时，只有被配置为输入的引脚才会导致该中断发生（即，当 RB2、RB3、RA0 和 RA1 中的任何一个引脚被配置为输出时，该引脚不再具有电平变化中断比较功能）。输入引脚（RB2、RB3、RA0 和 RA1）与上次读 PORTA 和 PORTB 锁存的旧值进行比较。这些引脚输出的“不匹配”值一起进行逻辑“或”运算，产生 RB 端口电平变化中断，并将标志位 RBIF（INTCON<0>）置 1。

此中断可将器件从休眠模式或任何空闲模式唤醒。用户可用以下方式在中断服务程序中清除该中断：

a) 读或写 PORTA 和 / 或 PORTB（ `MOVFF (ANY), PORTA` 和 `MOVFF (ANY), PORTB` 指令除外）。
b) 清零标志位 RBIF。

电平不匹配的状态将会一直不断地将RBIF标志位置1。而读 PORTA 和 PORTB 将结束该状态，并且允许将 RBIF 标志位清零。

建议使用此电平变化中断来实现按键唤醒以及其他使用 PORTA 和 PORTB 电平变化中断的应用。在使用电平变化中断功能时，建议不要查询 PORTA 和 PORTB 的状态。

表 **9-3**：　**PORTB I/O** 汇总

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RB0/PWM0	RB0	0	O	DIG	LATB<0> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<0> 数据输入；当 $\overline{RBPU}$ 位被清零时启用弱上拉。当使能模拟输入时被禁止。[1]
	PWM0	0	O	DIG	PWM 模块输出 PWM0。
RB1PWM1	RB1	0	O	DIG	LATB<1> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<1> 数据输入；当 $\overline{RBPU}$ 位被清零时启用弱上拉。当使能模拟输入时被禁止。[1]
	PWM1	0	O	DIG	PWM 模块输出 PWM1。
RB2/INT2/KBI2/CMP2/T1OSO/T1CKI	RB2	0	O	DIG	LATB<2> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<2> 数据输入；当 $\overline{RBPU}$ 位被清零时启用弱上拉。当使能模拟输入时被禁止。[1]
	INT2	1	I	ST	外部中断 2 输入。
	KBI2	1	I	TTL	电平变化中断引脚。
	CMP2	1	I	ANA	比较器 2 输入。
	T1OSO[2]	0	O	ANA	Timer1 振荡器输出。
	T1CKI[2]	1	I	ST	Timer1 时钟输入。
RB3/INT3/KBI3/CMP1/T1OSI	RB3	0	O	DIG	LATB<3> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<3> 数据输入；当 $\overline{RBPU}$ 位被清零时启用弱上拉。当使能模拟输入时被禁止。[1]
	INT3	1	I	ST	外部中断 3 输入。
	KBI3	1	I	TTL	电平变化中断引脚。
	CMP1	1	I	ANA	比较器 1 输入。
	T1OSI[2]	1	I	ANA	Timer1 振荡器输入。
RB4/PWM2	RB4	0	O	DIG	LATB<4> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<4> 数据输入；当 $\overline{RBPU}$ 位被清零时启用弱上拉。当使能模拟输入时被禁止。[1]
	PWM2	0	O	DIG	PWM 模块输出 PWM2。
RB5/PWM3	RB5	0	O	DIG	LATB<5> 数据输出。
		1	I	TTL	PORTB<5> 数据输入；当 $\overline{RBPU}$ 位清零时启用弱上拉。
	PWM3	0	O	DIG	PWM 模块输出 PWM3。
RB6/PWM4/PGC	RB6	0	O	DIG	LATB<6> 数据输出。
		1	I	TTL	PORTB<6> 数据输入；当 $\overline{RBPU}$ 位清零时启用弱上拉。
	PWM4	0	O	DIG	PWM 模块输出 PWM4。
	PGC	1	I	ST	在线调试器和 ICSP™ 编程时钟引脚。
RB7/PWM5/PGD	RB7	0	O	DIG	LATB<7> 数据输出。
		1	I	TTL	PORTB<7> 数据输入；当 $\overline{RBPU}$ 位清零时启用弱上拉。
	PWM5	0	O	TTL	PWM 模块输出 PWM4。
	PGD	0	O	DIG	在线调试器和 ICSP 编程数据引脚。

图注：　DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发器输入缓冲器； ANA = 模拟电平输入 / 输出；
x = 无关值 （TRIS 位不影响端口方向或者在此可被忽略）。

注　**1**：发生上电复位时的配置由 PBADEN 配置位决定。默认情况下 （PBADEN 置 1），这些引脚被配置为模拟输入；而当 PBADEN 清零时，这些引脚被配置为数字输入。

2：T1OSI 和 T1OSO/T1CKI 的设置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

表 9-4： 与 **PORTB** 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	44
LATB	PORTB 数据锁存器 （读取和写入数据锁存器）								43
TRISB	PORTB 数据方向控制寄存器								43
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	41
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	41
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	42

图注： — = 未用 （读为 0）。 PORTB 未使用阴影单元。

 超前信息

10.0 中断

PIC18F1230/1330 器件提供多个中断源及一个中断优先级功能，可以给大多数中断源分配高优先级或者低优先级。高优先级中断向量地址为 0008h，低优先级中断向量地址为 0018h。高优先级中断事件将中断所有可能正在进行的低优先级中断。

有 13 个寄存器用于控制中断的操作。它们是：

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1、 PIR2 和 PIR3
- PIE1、 PIE2 和 PIE3
- IPR1、 IPR2 和 IPR3

建议使用 MPLAB® IDE 提供的 Microchip 头文件命名这些寄存器中的位。这使得汇编器 / 编译器能够自动识别指定寄存器内这些位的位置。

通常，用三个位来控制中断源的操作。它们是：

- **标志位**表明发生了中断事件
- **允许位**在标志位置 1 时允许程序跳转到中断向量地址处执行
- **优先级位**用于选择是高优先级还是低优先级

通过将 IPEN 位 （RCON<7>）置 1，可使能中断优先级功能。当使能中断优先级时，有2位可允许全局中断。将 GIEH 位 （INTCON<7>）置 1，可允许所有优先级位置 1（高优先级）的中断。将 GIEL 位（INTCON<6>）置 1，可允许所有优先级位清零 （低优先级）的中断。当中断标志位、允许位以及相应的全局中断允许位均被置 1 时，程序将立即跳转到中断地址 0008h 或 0018h，具体地址取决于优先级位的设置。通过设置相应的允许位可以禁止单个中断。

当 IPEN 位被清零 （默认状态）时 , 便会禁止中断优先级功能，此时的中断与 PIC® 中档器件的相兼容。在兼容模式下，各个中断源的中断优先级位均不起作用。INTCON<6> 是 PEIE 位，它可允许 / 禁止所有的外设中断源。 INTCON<7> 是 GIE 位，它可允许 / 禁止所有的中断源。在兼容模式下，所有中断均跳转到地址 0008h。

当响应中断时， 全局中断允许位被清零以禁止其他中断。如果清零 IPEN 位，全局中断允许位就是 GIE 位。如果使用中断优先级，这个位将是 GIEH 位或者 GIEL 位。高优先级中断源会中断低优先级中断。处理高优先级中断时，低优先级中断将不被响应。

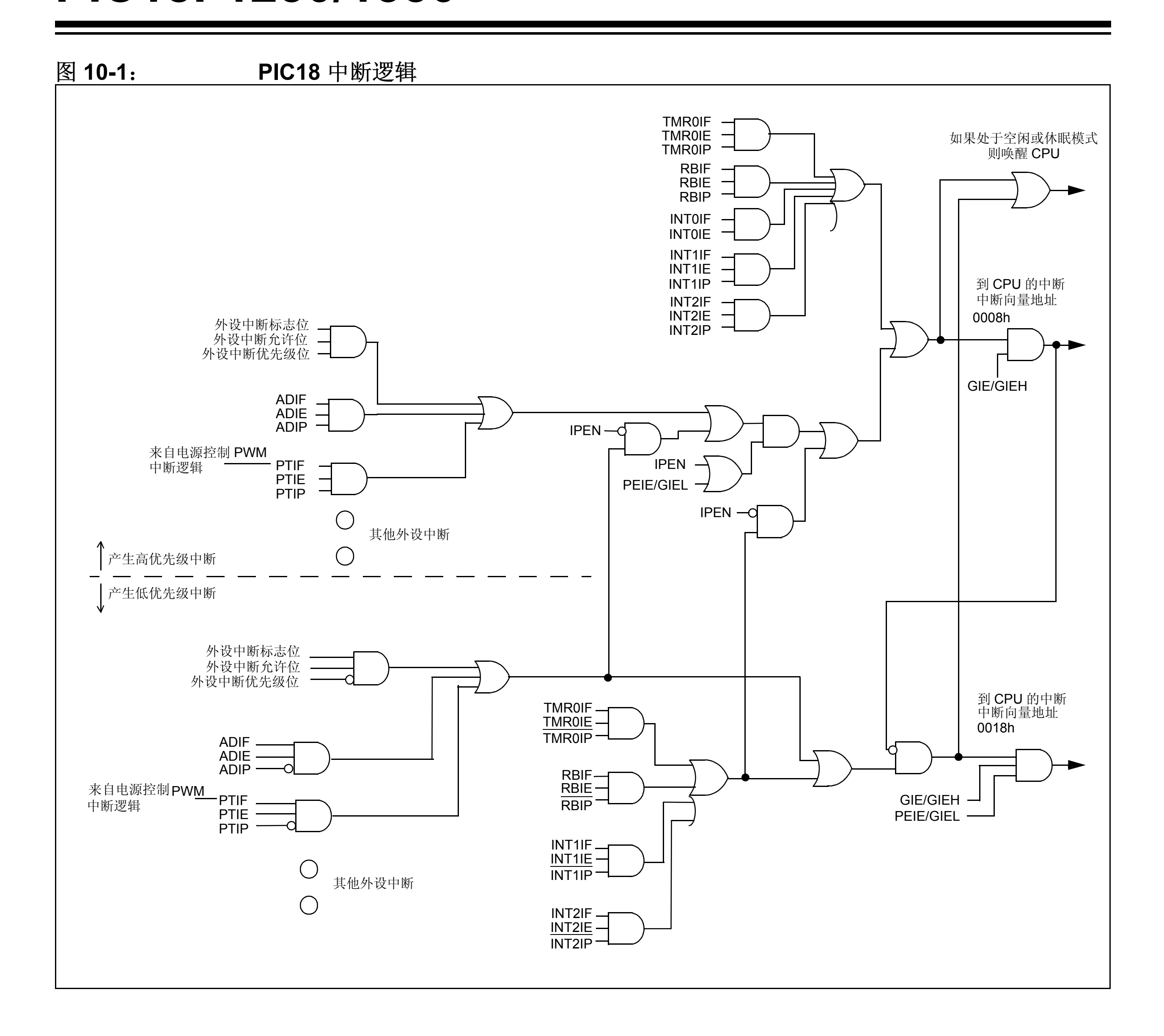
返回地址被压入堆栈， PC 装入中断向量地址 （0008h 或 0018h）。进入中断服务程序之后，就可以通过查询中断标志位来确定中断源。在重新允许中断前，必须用软件将中断标志位清零，以避免重复响应该中断。

执行 “中断返回”指令 `RETFIE`，退出中断程序并将 GIE 位 （若使用中断优先级，则为 GIEH 或 GIEL 位）置 1 以重新允许中断。

对于外部中断事件，诸如 INT 引脚中断或者 PORTB 输入电平变化中断，中断响应延时将会是 3 到 4 个指令周期。对于单周期或双周期指令，中断响应延时完全相同。不管对应的中断允许位和 GIE 位状态如何，中断标志位均被置 1。

注：	当中断被允许时，不要使用 `MOVFF` 指令来修改中断控制寄存器。否则可能引起单片机操作出错。

图 10-1： PIC18 中断逻辑



 超前信息

10.1 INTCON 寄存器

INTCON 寄存器是可读写的寄存器，包含多个允许位、优先级位和标志位。

注：	当中断条件产生时，不管相应的中断允许位或全局允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将该中断标志位清零。故中断标志位可以用软件查询。

寄存器 10-1： INTCON：中断控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **GIE/GIEH**：全局中断允许位
当 IPEN = 0 时：
1 = 允许所有未屏蔽的中断
0 = 禁止所有中断
当 IPEN= 1 时：
1 = 允许所有高优先级中断
0 = 禁止所有中断

bit 6 **PEIE/GIEL**：外设中断允许位
当 IPEN= 0 时：
1 = 允许所有未屏蔽的外设中断
0 = 禁止所有外设中断
当 IPEN= 1 时：
1 = 允许所有低优先级的外设中断
0 = 禁止所有低优先级的外设中断

bit 5 **TMR0IE**：TMR0 溢出中断允许位
1 = 允许 TMR0 溢出中断
0 = 禁止 TMR0 溢出中断

bit 4 **INT0IE**：INT0 外部中断允许位
1 = 允许 INT0 外部中断
0 = 禁止 INT0 外部中断

bit 3 **RBIE**：RB 端口电平变化中断允许位
1 = 允许 RB 端口电平变化中断
0 = 禁止 RB 端口电平变化中断

bit 2 **TMR0IF**：TMR0 溢出中断标志位
1 = TMR0 寄存器已经溢出 （必须由软件清零）
0 = TMR0 寄存器未溢出

bit 1 **INT0IF**：INT0 外部中断标志位
1 = 发生了 INT0 外部中断 （必须由软件清零）
0 = 未发生 INT0 外部中断

bit 0 **RBIF**：RB 端口电平变化中断标志位 (1)
1 = RB7:RB4 引脚中至少有一个引脚的电平状态发生了改变 （必须由软件清零）
0 = RB7:RB4 引脚的电平状态没有改变

注 1： 引脚上电平变化会一直不断地将此位置 1。读取 PORTB 可以结束这种情况，并允许将该位清零。

寄存器 10-2： INTCON2：中断控制寄存器 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **$\overline{\text{RBPU}}$**：PORTB 上拉使能位
1 = 禁止所有 PORTB 上拉
0 = 按各个端口锁存值使能 PORTB 上拉

bit 6 **INTEDG0**：外部中断 0 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 5 **INTEDG1**：外部中断 1 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 4 **INTEDG2**：外部中断 2 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 3 **INTEDG3**：外部中断 3 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 2 **TMR0IP**：TMR0 溢出中断优先级位
1 = 高优先级
0 = 低优先级

bit 1 **INT3IP**：INT3 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 0 **RBIP**：RB 端口电平变化中断优先级位
1 = 高优先级
0 = 低优先级

注：	当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断前，确保先将该中断标志位清零。故中断标志位可以用软件查询。

寄存器 10-3： INTCON3：中断控制寄存器 3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **INT2IP**：INT2 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 6 **INT1IP**：INT1 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 5 **INT3IE**：INT3 外部中断允许位
1 = 允许 INT3 外部中断
0 = 禁止 INT3 外部中断

bit 4 **INT2IE**：INT2 外部中断允许位
1 = 允许 INT2 外部中断
0 = 禁止 INT2 外部中断

bit 3 **INT1IE**：INT1 外部中断允许位
1 = 允许 INT1 外部中断
0 = 禁止 INT1 外部中断

bit 2 **INT3IF**：INT3 外部中断标志位
1 = 发生了 INT3 外部中断 （必须由软件清零）
0 = 未发生 INT3 外部中断

bit 1 **INT2IF**：INT2 外部中断标志位
1 = 发生了 INT2 外部中断 （必须由软件清零）
0 = 未发生 INT2 外部中断

bit 0 **INT1IF**：INT1 外部中断标志位
1 = 发生了 INT1 外部中断 （必须由软件清零）
0 = 未发生 INT1 外部中断

注：	当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断前，确保先将该中断标志位清零。故中断标志位可以用软件查询。

10.2 PIR 寄存器

PIR 寄存器包含各外设中断的标志位。根据外设中断源的数量，总共有 3 个外设中断请求 （标志）寄存器（PIR1、 PIR2 和 PIR3）。

> 注 1：当中断条件产生时，不管相应的中断允许位或全局允许位 GIE（INTCON<7>）的状态如何，中断标志位都将置 1。
>
> 2：用户软件应在允许一个中断之前，确保先将该中断标志位清零；同时在响应该中断后，也应该将该中断标志位清零。

寄存器 10-4： PIR1：外设中断请求 （标志）寄存器 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 未用：读为 0

bit 6 **ADIF**：A/D 转换器中断标志位
1 = 一次 A/D 转换已完成 （必须由软件清零）
0 = A/D 转换未完成

bit 5 **RCIF**：EUSART 接收中断标志位
1 = EUSART 接收缓冲器 RCREG 已满 （当读取 RCREG 时清零）
0 = EUSART 接收缓冲器为空

bit 4 **TXIF**：EUSART 发送中断标志位
1 = EUSART 发送缓冲器 TXREG 为空 （当写入 TXREG 时清零）
0 = EUSART 发送缓冲器已满

bit 3 **CMP2IF**：模拟比较器 2 中断标志位
1 = 自上次读取后， CMP2 的输出已发生了更改
0 = 自上次读取后， CMP2 的输出未发生更改

bit 2 **CMP1IF**：模拟比较器 1 中断标志位
1 = 自上次读取后， CMP1 的输出已发生了更改
0 = 自上次读取后， CMP1 的输出未发生更改

bit 1 **CMP0IF**：模拟比较器 0 中断标志位
1 = 自上次读取后， CMP0 的输出已发生了更改
0 = 自上次读取后， CMP0 的输出未发生更改

bit 0 **TMR1IF**：TMR1 溢出中断标志位
1 = TMR1 寄存器已发生溢出 （必须由软件清零）
0 = TMR1 寄存器未发生溢出

寄存器 10-5： PIR2：外设中断请求 （标志）寄存器 2

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	U-0	U-0
OSCFIF	—	—	EEIF	—	LVDIF	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **OSCFIF**：振荡器失效中断标志位
1 = 器件振荡器发生故障，改由 INTOSC 作为时钟输入 （必须由软件清零）
0 = 器件时钟正常运行

bit 6-5 未用：读为 0

bit 4 **EEIF**：数据 EEPROM/ 闪存写操作中断标志位
1 = 写操作完成 （必须由软件清零）
0 = 写操作未完成或还未开始

bit 3 未用：读为 0

bit 2 **LVDIF**：低电压检测中断标志位
1 = 发生了低电压条件
0 = 未发生低电压条件

bit 1-0 未用：读为 0

寄存器 10-6： PIR3：外设中断请求 （标志）寄存器 3

U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
—	—	—	PTIF	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-5 未用：读为 0

bit 4 **PTIF**：PWM 时基中断位
1 = PWM 时基与 PTPER 寄存器中的值相匹配。根据后分频器的设置产生中断。PTIF 必须由软件清零。
0 = PWM 时基与 PTPER 寄存器中的值不匹配

bit 3-0 未用：读为 0

10.3 PIE 寄存器

PIE 寄存器包含各外设中断的允许位。根据外设中断源的数量，总共有 3 个外设中断允许寄存器（PIE1、PIE2 和 PIE3）。当 IPEN = 0 时，要允许任何外设中断就必须将 PEIE 位置 1。

寄存器 10-7： PIE1：外设中断允许寄存器 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 未用：读为 0

bit 6 **ADIE**：A/D 转换器中断允许位
1 = 允许 A/D 中断
0 = 禁止 A/D 中断

bit 5 **RCIE**：EUSART 接收中断允许位
1 = 允许 EUSART 接收中断
0 = 禁止 EUSART 接收中断

bit 4 **TXIE**：EUSART 发送中断允许位
1 = 允许 EUSART 发送中断
0 = 禁止 EUSART 发送中断

bit 3 **CMP2IE**：模拟比较器 2 中断允许位
1 = 允许 CMP2 中断
0 = 禁止 CMP2 中断

bit 2 **CMP1IE**：模拟比较器 1 中断允许位
1 = 允许 CMP1 中断
0 = 禁止 CMP1 中断

bit 1 **CMP0IE**：模拟比较器 0 中断允许位
1 = 允许 CMP0 中断
0 = 禁止 CMP0 中断

bit 0 **TMR1IE**：TMR1 溢出中断允许位
1 = 允许 TMR1 溢出中断
0 = 禁止 TMR1 溢出中断

 超前信息

寄存器 10-8： PIE2：外设中断允许寄存器 2

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	U-0	U-0
OSCFIE	—	—	EEIE	—	LVDIE	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **OSCFIE**：振荡器失效中断允许位
1 = 允许
0 = 禁止

bit 6-5 未用：读为 0

bit 4 **EEIF**：数据 EEPROM/ 闪存写操作中断允许位
1 = 允许
0 = 禁止

bit 3 未用：读为 0

bit 2 **LVDIE**：低电压检测中断允许位
1 = 允许
0 = 禁止

bit 1-0 未用：读为 0

寄存器 10-9： PIE3：外设中断允许寄存器 3

U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
—	—	—	PTIE	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-5 未用：读为 0

bit 4 **PTIE**：PWM 时基中断允许位
1 = 允许 PWM 中断
0 = 禁止 PWM 中断

bit 3-0 未用：读为 0

10.4 IPR 寄存器

IPR 寄存器包含各外设中断的优先级位。根据外设中断源的数量，总共有 3 个外设中断优先级寄存器（IPR1、IPR2 和 IPR3）。使用优先级位要求将中断优先级使能（IPEN）位置 1。

寄存器 10-10： IPR1：外设中断优先级寄存器 1

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 未用：读为 0

bit 6 **ADIP**：A/D 转换器中断优先级位
1 = 高优先级
0 = 低优先级

bit 5 **RCIP**：EUSART 接收中断优先级位
1 = 高优先级
0 = 低优先级

bit 4 **TXIP**：EUSART 发送中断优先级位
1 = 高优先级
0 = 低优先级

bit 3 **CMP2IP**：模拟比较器 2 中断优先级位
1 = 高优先级
0 = 低优先级

bit 2 **CMP1IP**：模拟比较器 1 中断优先级位
1 = 高优先级
0 = 低优先级

bit 1 **CMP0IP**：模拟比较器 0 中断优先级位
1 = 高优先级
0 = 低优先级

bit 0 **TMR1IP**：TMR1 溢出中断优先级位
1 = 高优先级
0 = 低优先级

超前信息

寄存器 10-11： IPR2：外设中断优先级寄存器 2

R/W-1	U-0	U-0	R/W-1	U-0	R/W-1	U-0	U-0
OSCFIP	—	—	EEIP	—	LVDIP	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **OSCFIP**：振荡器失效中断优先级位
1 = 高优先级
0 = 低优先级

bit 6-5 未用：读为 0

bit 4 **EEIP**：数据 EEPROM/ 闪存写操作中断优先级位
1 = 高优先级
0 = 低优先级

bit 3 未用：读为 0

bit 2 **LVDIP**：低电压检测中断优先级位
1 = 高优先级
0 = 低优先级

bit 1-0 未用：读为 0

寄存器 10-12： IPR3：外设中断优先级寄存器 3

U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
—	—	—	PTIP	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-5 未用：读为 0

bit 4 **PTIP**：PWM 时基中断优先级位
1 = 高优先级
0 = 低优先级

bit 3-0 未用：读为 0

10.5 RCON 寄存器

RCON 寄存器包含几个标志位，可以用来确定器件上次复位或从空闲或休眠模式被唤醒的原因。RCON还包含 IPEN 位，该位可以使能中断优先级。

在第 **4.1 节“RCON 寄存器”**中对 SBOREN 位和复位标志位的操作做了更详细地讨论。

寄存器 10-13： RCON：复位控制寄存器

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **IPEN：**中断优先级使能位
1 = 使能中断优先级
0 = 禁止中断优先级 （PIC16CXXX 兼容模式）

bit 6 **SBOREN：**软件 BOR 使能位 (1)
欲知位操作的详细信息，请参见寄存器 4-1。

bit 5 **未用：**读为 0

bit 4 **$\overline{\text{RI}}$：**RESET 指令标志位
欲知位操作的详细信息，请参见寄存器 4-1。

bit 3 **$\overline{\text{TO}}$：**看门狗定时器超时溢出标志位
欲知位操作的详细信息，请参见寄存器 4-1。

bit 2 **$\overline{\text{PD}}$：**掉电检测标志位
欲知位操作的详细信息，请参见寄存器 4-1。

bit 1 **$\overline{\text{POR}}$：**上电复位状态位 (2)
欲知位操作的详细信息，请参见寄存器 4-1。

bit 0 **$\overline{\text{BOR}}$：**欠压复位状态位
欲知位操作的详细信息，请参见寄存器 4-1。

注 **1：** 如果使能 SBOREN ，该位的复位状态为 1，否则为 0。更多信息请参见寄存器 4-1。
2： $\overline{\text{POR}}$ 的实际复位值取决于器件复位的类型。更多信息请参见寄存器 4-1。

10.6 INTn 引脚中断

RA0/INT0、RA1/INT1、RB2/INT2 和 RB3/INT3 引脚的外部中断是边沿触发的。如果 INTCON2 寄存器中对应的 INTEDGx 位置 1 （= 1），则该中断由上升沿触发；如果该位清零，则中断由下降沿触发。当引脚上出现一个有效边沿时，对应的标志位 INTxIF 被置 1。通过清零对应的允许位 INTxIE，可禁止该中断。在重新允许该中断前，必须在中断服务程序中先用软件将 INTxIF 中断标志位清零。

如果 INTxIE 位在进入空闲或休眠模式前被置 1，则所有的外部中断（INT0、INT1、INT2 和 INT3）均能将处理器唤醒。如果全局中断允许位 GIE 被置 1，则处理器将在被唤醒之后跳转到中断向量处执行程序。

INT1、 INT2 和 INT3 的中断优先级由中断优先级位 INT1IP（INTCON3<6>）、INT2IP（INTCON3<7>）和 INT3IP （INTCON2<1>）包含的值决定。没有与 INT0 相关的优先级位。INT0 始终是一个高优先级的中断源。

10.7 TMR0 中断

在 8 位模式下 （默认设置）， TMR0 寄存器的溢出（FFh → 00h)）将会使标志位 TMR0IF 置 1。在 16 位模式下， TMR0H:TMR0L 寄存器对的溢出 （FFFFh → 0000h）会使 TMR0IF 标志位置 1。通过将允许位 TMR0IE （INTCON<5>）置 1 或清零，可以允许或禁止该中断。 Timer0 的中断优先级由中断优先级位 TMR0IP （INTCON2<2>）包含的值决定。欲进一步了解 Timer0 模块的详细内容，请参见**第 11.0 节“Timer0 模块”**。

10.8 电平变化中断

PORTA<1:0> 和 / 或 PORTB<2:3> 上的输入电平变化会将标志位 RBIF （INTCON<0>）置 1。通过将允许位 RBIE （INTCON<3>）置 1 或清零，可以允许或禁止该中断。电平变化中断的优先级由中断优先级位 RBIP （INTCON2<0>）包含的值决定。

10.9 中断的现场保护

在中断期间，返回的 PC 地址被压入堆栈。另外，WREG、 STATUS 以及 BSR 寄存器的值被压入快速返回堆栈。如果未使用从中断快速返回功能（见**第 5.3 节“数据存储器构成”**），用户可能需要在进入中断服务程序时，保存 WREG、 STATUS 以及 BSR 寄存器的值。根据用户的具体应用，可能还需要保存其他寄存器的值。例 10-1 在执行中断服务程序期间，保存并恢复 WREG、 STATUS 和 BSR 寄存器的值。

例 10-1： 将 STATUS、 WREG 和 BSR 寄存器的值保存在 RAM 中

```
MOVWF   W_TEMP                      ; W_TEMP is in virtual bank
MOVFF   STATUS, STATUS_TEMP         ; STATUS_TEMP located anywhere
MOVFF   BSR, BSR_TEMP               ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF   BSR_TEMP, BSR               ; Restore BSR
MOVF    W_TEMP, W                   ; Restore WREG
MOVFF   STATUS_TEMP, STATUS         ; Restore STATUS
```

注：

11.0 TIMER0 模块

Timer0 模块具有以下特征：

- 可通过软件选择，作为 8 位或 16 位定时器 / 计数器
- 可读写的寄存器
- 专用的 8 位软件可编程预分频器
- 可选的时钟源 （内部或外部）
- 8 位模式下 FFh 到 00h 的溢出中断， 16 位模式下 FFFFh 到 0000h 的溢出中断
- 外部时钟的边沿选择

图 11-1 给出了 8 位模式下 Timer0 模块的简化框图，图 11-2 给出了 16 位模式 Timer0 模块的简化框图。

T0CON寄存器（寄存器 11-1）控制该模块的工作方式，包括预分频比的选择。该寄存器是可读写的。

寄存器 11-1： T0CON：TIMER0 控制寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **TMR0ON**：Timer0 开 / 关控制位
1 = 使能 Timer0
0 = 禁止 Timer0

bit 6 **T016BIT**：Timer0 16 位控制位
1 = Timer0 被配置为 8 位定时器 / 计数器
0 = Timer0 被配置为 16 位定时器 / 计数器

bit 5 **T0CS**：Timer0 时钟源选择位
1 = T0CKI 引脚上的传输信号作为时钟源
0 = 内部指令周期时钟 （CLKO）作为时钟源

bit 4 **T0SE**：Timer0 时钟源边沿选择位
1 = 在 T0CKI 引脚上信号的下降沿触发递增
0 = 在 T0CKI 引脚上信号的上升沿触发递增

bit 3 **PSA**：Timer0 预分频器分配位
1 = 未分配 Timer0 预分频器。 Timer0 时钟输入不经过预分频器。
0 = 已分配 Timer0 预分频器。 Timer0 时钟输入信号来自预分频器的输出。

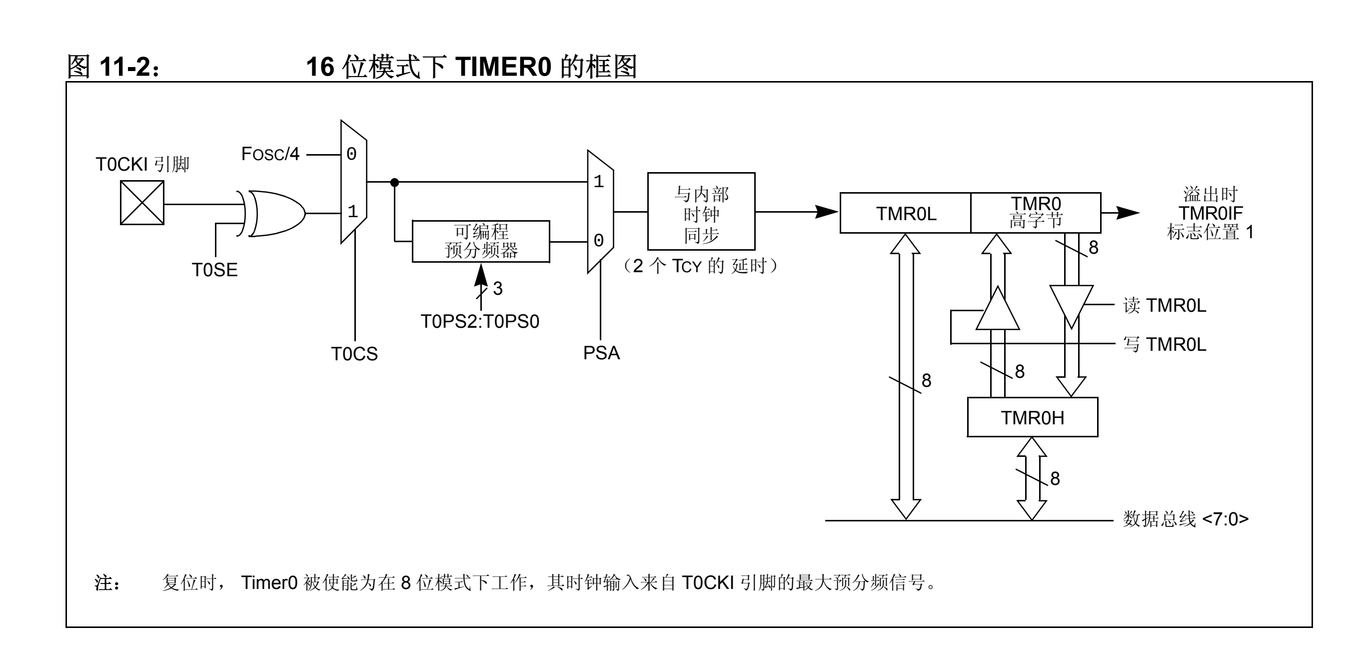
bit 2-0 **T0PS2:T0PS0**：Timer0 预分频比选择位
111 = 1:256 预分频比
110 = 1:128 预分频比
101 = 1:64 预分频比
100 = 1:32 预分频比
011 = 1:16 预分频比
010 = 1:8 预分频比
001 = 1:4 预分频比
000 = 1:2 预分频比

图 11-1： 8 位模式下 TIMER0 的框图

FOSC/4
T0CKI 引脚
0
1
T0SE
T0CS
可编程
预分频器
3
T0PS2:T0PS0
1
0
PSA
与内部
时钟
同步
（2 个 TCY 的 延时）
数据总线
8
TMR0
溢出时 TMR0IF
标志位置 1

注： 复位时， Timer0 被使能为在 8 位模式下工作，其时钟输入来自 T0CKI 引脚的最大预分频信号。

图 11-2： 16 位模式下 TIMER0 的框图



注： 复位时， Timer0 被使能为在 8 位模式下工作，其时钟输入来自 T0CKI 引脚的最大预分频信号。

 超前信息

11.1 Timer0 工作原理

Timer0 既可用作定时器亦可用作计数器。

具体的模式由 T0CS 位选择。在定时器模式下，Timer0 模块的计数在每个指令周期都会加 1（不使用预分频器）。如果写入 TMR0 寄存器，那么在随后的两个指令周期内，计数将不再递增。用户可通过将校正后的值写入 TMR0 寄存器避开上述问题。

通过将 T0CS 位置 1 选择计数器模式。在计数器模式下，Timer0 可在 RA4/T0CKI/AN2/VREF+ 引脚信号的每个上升沿或下降沿递增。触发递增的边沿由 Timer0 时钟源边沿选择位T0SE决定。清零此位选择上升沿递增。

可以使用外部时钟源来驱动 Timer0，但是必须确保外部时钟与内部时钟（Tosc）相位同步。在同步之后，定时器 / 计数器仍需要一定的延时才会引发递增操作。

11.2 预分频器

Timer0 模块的预分频器为一个 8 位计数器。此预分频器不可直接读写。

PSA 和 T0PS2:T0PS0 位决定预分频器的分配和预分频比值。

将 PSA 位清零可将预分频器分配给 Timer0 模块。预分频比可以在 1:2、 1:4 直到 1:256 之间选择。

若将预分频器分配给 Timer0 模块，所有以 TMR0 寄存器为写入对象的指令（如 `CLRF TMR0`、`MOVWF TMR0` 和 `BSF TMR0, x` 等）都将使预分频器的计数值清零。

注：	若将预分频器分配给 Timer0，写入 TMR0 会将预分频器的计数值清零，但不会改变预分频器的分配。

11.2.1 切换预分频器的分配

预分频器的分配完全由软件控制（即，它可在程序执行期间被随时更改）。

11.3 Timer0 中断

当 TMR0 寄存器发生溢出时（8 位模式下，从 FFh 到 00h ；或 16 位模式下，从 FFFFh 到 0000h），将产生 TMR0 中断。这种溢出会将标志位 TMR0IF 置 1。可以通过清零 TMR0IE 位来屏蔽此中断。在重新允许该中断前，必须在中断服务程序中用软件清零 TMR0IF 位。由于定时器需要时钟才能正常工作，因此在休眠模式下即便 T0CS 置 1， TMR0 中断也无法将处理器唤醒。

11.4 16 位定时器读写模式

TMR0H 并不是 16 位模式下 Timer0 的高字节，而是 Timer0 高字节的缓冲寄存器（见图 11-2），Timer0 的高字节是不可以被直接读写的。在读 TMR0L 时使用 Timer0 高字节的内容更新 TMR0H。这样可以一次读取 Timer0 的全部 16 位，而无需验证读到的高字节和低字节的有效性（在高、低字节分两次连续读取的情况下，由于可能存在进位，因此需要验证读到字节的有效性）。

同样，写入 Timer0 的高字节也是通过 TMR0H 缓冲寄存器来操作的。在写入 TMR0L 的同时，使用 TMR0H 的内容更新 Timer0 的高字节。这样一次就可以完成 Timer0 全部 16 位的更新。

表 11-1： 与 TIMER0 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
TMR0L	Timer0 寄存器的低字节								42
TMR0H	Timer0 寄存器的高字节								42
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
T0CON	TMR0ON	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	42
TRISA	RA7[(1)]	RA6[(1)]	PORTA 数据方向控制寄存器						43

图注： x = 未知，u = 不变，— = 未用（读为 0）。 Timer0 模块不使用阴影单元。

注 1： 根据在 CONFIG1H 中选择的振荡器模式，可将 RA6 和 RA7 使能为 I/O 引脚。

注：

12.0 TIMER1 模块

Timer1 定时器 / 计数器模块具有以下特征：

- 16 位定时器 / 计数器（两个 8 位寄存器： TMR1H 和 TMR1L）
- 可读写 （以上两个寄存器）
- 内部或外部时钟选择
- 从 FFFFh 到 0000h 的溢出中断
- 系统时钟的工作状态

图 12-1 是 Timer1 模块的简化框图。

寄存器 12-1 详细说明了 Timer1 控制寄存器。此寄存器控制 Timer1 模块的工作模式，并且包含 Timer1 振荡器使能位 （T1OSCEN） 。可以通过将控制位 TMR1ON（T1CON<0>）置 1 或清零来使能或禁止 Timer1。

Timer1 振荡器可以用作功耗管理模式下的辅助时钟源。当 T1RUN 位被置 1 时，由 Timer1 振荡器提供系统时钟。如果使能了故障保护时钟监视器并且 Timer1 振荡器在提供时钟信号时发生了故障，查询 T1RUN 位可以确定时钟源是由 Timer1 振荡器提供还是由其他时钟源提供。

利用较少的外部元件和代码，Timer1 可为应用提供实时时钟 （RTC）。

寄存器 12-1： T1CON：TIMER1 控制寄存器

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **RD16**：16 位读 / 写模式使能位
1 = 使能通过一次 16 位操作对 Timer1 寄存器进行读写
0 = 使能通过两次 8 位操作对 Timer1 寄存器进行读写

bit 6 **T1RUN**：Timer1 系统时钟状态位
1 = 器件时钟由 Timer1 振荡器产生
0 = 器件时钟由另一个时钟源产生

bit 5-4 **T1CKPS1:T1CKPS0**：Timer1 输入时钟预分频比选择位
11 = 1:8 预分频比
10 = 1:4 预分频比
01 = 1:2 预分频比
00 = 1:1 预分频比

bit 3 **T1OSCEN**：Timer1 振荡器使能位
1 = 使能 Timer1 振荡器
0 = 关闭 Timer1 振荡器
为了消除功率泄漏，关断了振荡器反相器和反馈电阻。

bit 2 **$\overline{\text{T1SYNC}}$**：Timer1 外部时钟输入同步选择位
当 TMR1CS = 1 时：
1 = 不与外部时钟输入同步
0 = 与外部时钟输入同步
当 TMR1CS = 0 时：
忽略此位。当 TMR1CS = 0 时， Timer1 使用内部时钟。

bit 1 **TMR1CS**：Timer1 时钟源选择位
1 = 使用 T1OSO/T1CKI 引脚上的外部时钟 （上升沿触发计数）**(1)**
0 = 内部时钟 （FOSC/4）

bit 0 **TMR1ON**：Timer1 使能位
1 = 使能 Timer1
0 = 禁止 Timer1

注 **1**：T1OSI 和 T1OSO/T1CKI 的设置取决于 CONFIG3H 中 T1OSCMX 配置位的值。

12.1 Timer1 工作原理

Timer1 可工作在以下模式：

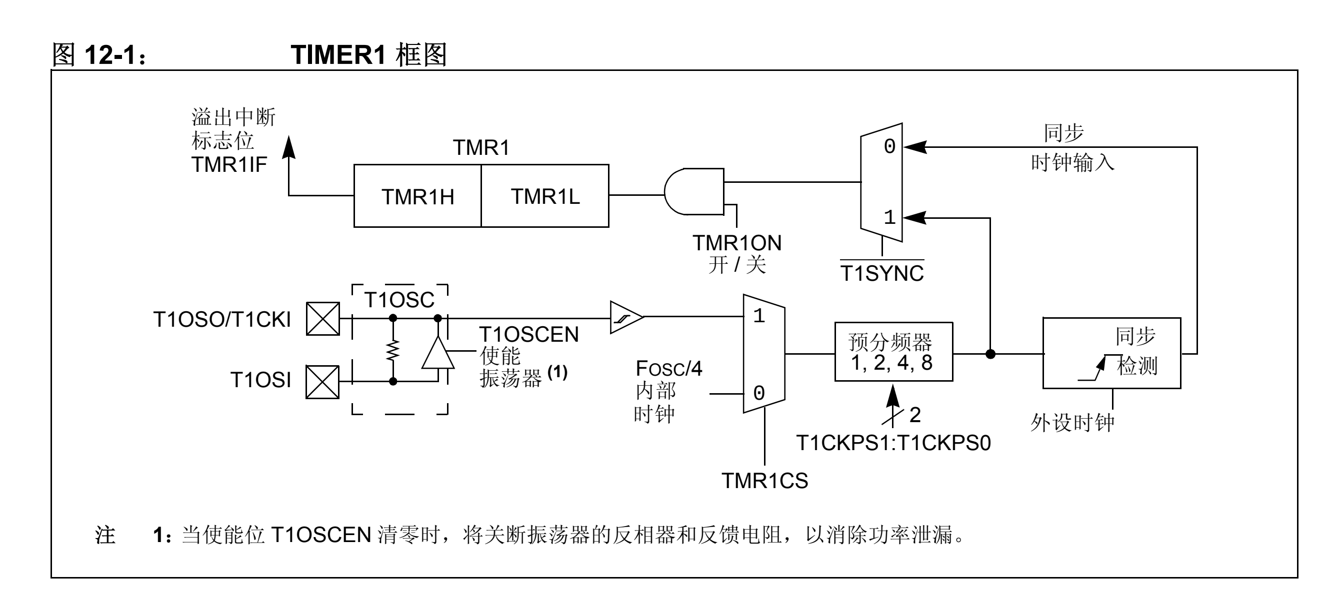
- 定时器
- 同步计数器
- 异步计数器

工作模式由时钟选择位 TMR1CS（T1CON<1>）决定。

当 TMR1CS = 0 时，Timer1 在每个内部指令周期计时 / 计数递增。当 TMR1CS = 1 时，Timer1 在 Timer1 外部时钟输入信号或 Timer1 振荡器信号（如果使能）的每个上升沿计时 / 计数递增。

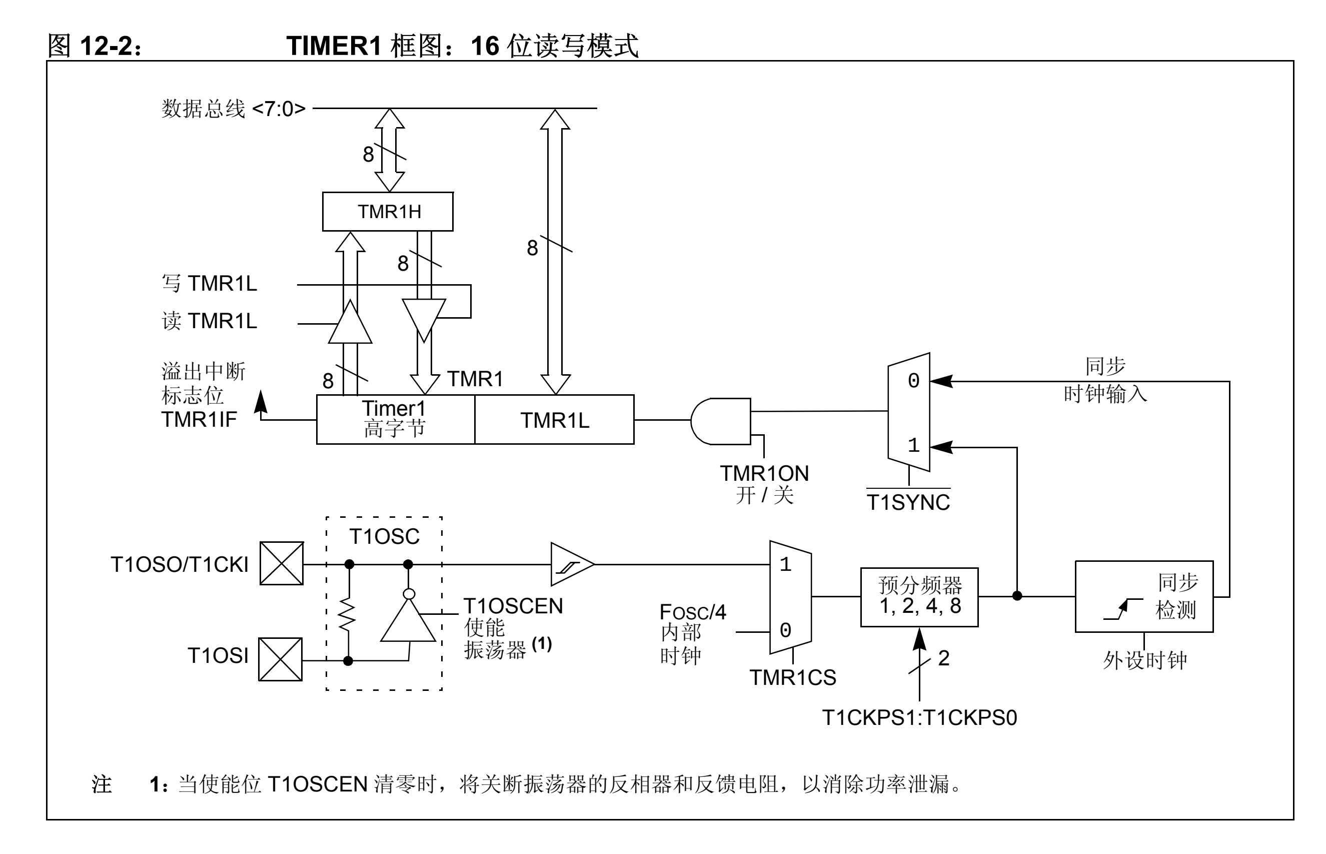
当使能 Timer1 振荡器（T1OSCEN 置 1）时， T1OSI 和 T1OSO/T1CKI 引脚变为输入引脚。这意味着相应的 TRISA 位的值被忽略并且这些引脚的读取值为 0。

图 12-1： TIMER1 框图



注 1：当使能位 T1OSCEN 清零时，将关断振荡器的反相器和反馈电阻，以消除功率泄漏。

图 12-2： TIMER1 框图：16 位读写模式



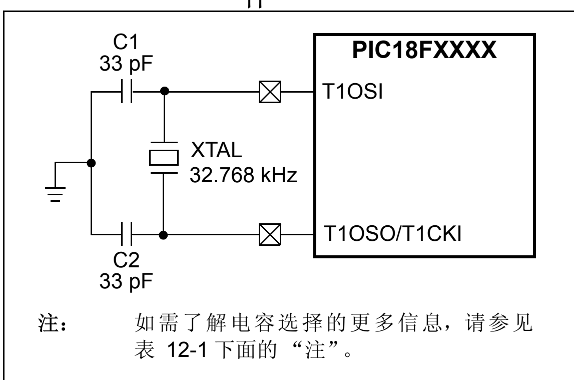
注 1：当使能位 T1OSCEN 清零时，将关断振荡器的反相器和反馈电阻，以消除功率泄漏。

12.2 Timer1 振荡器

在芯片内部 T1OSI （输入）引脚和 T1OSO/TICKI （放大器输出）引脚之间连接了晶振电路。这些引脚的设置取决于配置位 T1OSCMX 的值 （见**第 19.1 节 “配置位”**）。通过将控制位 T1OSCEN（T1CON<3>）置 1 可使能该振荡电路。此振荡电路是一种低功耗电路，它采用了额定振荡频率为 32 kHz 的晶振，在所有的功耗管理模式下都可继续运行。图 12-3 所示为典型的 LP 振荡器电路。表 12-1 显示了 Timer1 振荡器的电容选择。

用户必须提供软件延时来确保 Timer1 振荡器的正常起振。

图 12-3： TIMER1 LP 振荡器的外部元件



注： 如需了解电容选择的更多信息，请参见表 12-1 下面的 “注”。

表 12-1： Timer1 振荡器的电容选择

振荡器类型	频率	C1	C2
LP	32 kHz	27 pF(1)	27 pF(1)

注 1：Microchip 建议在验证振荡电路时从这些值开始。
2：选用较大的电容值虽然可以提高振荡器的稳定性，但同时也会延长起振时间。
3：由于谐振器 / 晶振的特性各不相同，因此用户应当向谐振器 / 晶振制造厂商咨询外部元件的相应值。
4：上述电容值仅供设计参考。

12.2.1 使用 TIMER1 作为时钟源

在功耗管理模式中也可以将Timer1振荡器用作时钟源。通过将时钟选择位 SCS1:SCS0 （OSCCON<1:0>）设置为 01，器件可以切换到 SEC_RUN 模式，此时 CPU 和外设均使用 Timer1 振荡器作为时钟源。如果 IDLEN 位 （OSCCON<7>）被清零并且执行了 SLEEP 指令，器件将进入 SEC_IDLE 模式。欲知更多详情，请参见**第 3.0 节 “功耗管理模式”**。

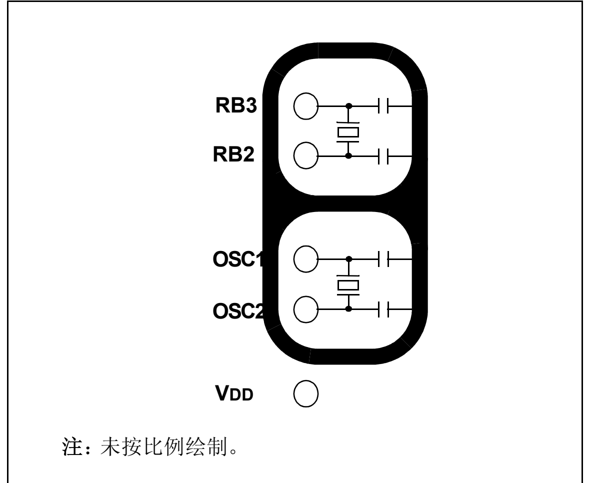
只要 Timer1 振荡器提供器件时钟，Timer1 系统时钟状态标志位 T1RUN （T1CON<6>）就会置 1。这可用于确定控制器的当前时钟模式。该位也可指示故障保护时钟监视器当前正使用的时钟源。如果使能了故障保护时钟监视器并且 Timer1 振荡器在提供时钟信号时发生了故障，查询 T1RUN 位可以确定时钟源是 Timer1 振荡器还是其他时钟源。

12.3 Timer1 振荡器布线注意事项

如图 12-3所示，振荡器电路应该尽可能靠近单片机。除了 VSS 或 VDD 外，在振荡电路区域内不应有其他电路。

如果必须要在该振荡器附近布置高速电路（如 PWM 引脚，或使用 OSC2 引脚的主振荡器），那么在该振荡电路周围布置接地保护环 （如图 12-4 所示），对于单面 PCB 板或外加接地层的 PCB 板来讲可能会有帮助。

图 12-4： 带有接地保护环的振荡电路



注：未按比例绘制。

12.4 Timer1 中断

TMR1 寄存器对（TMR1H:TMR1L）从 0000h 开始递增，直到 FFFFh，然后溢出从 0000h 重新开始计数。如果允许 Timer1 中断，该中断就会在溢出时产生，并将中断标志位 TMR1IF（PIR1<0>）置 1。可以通过将 Timer1 中断允许位 TMR1IE（PIE1<0>）置 1 或清零来允许或禁止该中断。

12.5 Timer1 16 位读 / 写模式

可将 Timer1 配置为 16 位读写模式（见图 12-2）。当 RD16 控制位（T1CON<7>）置 1 时，TMR1H 的地址被映射到 Timer1 的高字节缓冲寄存器。对 TMR1L 的读操作将把 Timer1 的高字节内容装入 Timer1 高字节缓冲寄存器。这种方式使用户可以精确地读取 Timer1 的全部 16 位，而不需要像先读高字节再读低字节那样由于两次读取之间可能存在进位，而不得不验证读取的有效性。

对 Timer1 的高字节进行写操作也必须通过 TMR1H 缓冲器进行。在写入 TMR1L 的同时，使用 TMR1H 的内容更新 Timer1 的高字节。这样允许用户将 16 位值一次写入 Timer1 的高字节和低字节。

在该模式下不能直接读写 Timer1 的高字节。所有读写都必须通过 Timer1 高字节缓冲器进行。写入 TMR1H 不会清零 Timer1 预分频器。只有在写 TMR1L 时才会清零预分频器。

12.6 使用 Timer1 作为实时时钟

为 Timer1 添加外部 LP 振荡器（如**第 12.2 节“Timer1 振荡器”**中所述），可以为用户提供 RTC 功能。这是通过一个提供精确时基的廉价时钟晶振以及几行计算时间的应用程序代码实现的。当器件工作于休眠模式下并使用电池或超大容量电容作为电源时，可以省去额外的 RTC 器件和备用电池。

应用代码程序 `RTCisr`（如例 12-1 所示），说明了使用中断服务程序以 1 秒的间隔递增计数器的简单方法。将 TMR1 寄存器对的值不断加 1 直至溢出，触发中断并调用中断服务程序，该程序会使秒计数器加 1，而分钟和小时计数器则会在前面的计数器溢出时加 1。

由于这对寄存器为 16 位宽，因此使用 32.768 kHz 时钟，将其计数到溢出需要 2 秒。 要使溢出按所需的 1 秒间隔进行，必须预先装载这对寄存器。最简单的方法是使用 BSF 指令将 TMR1H 的最高有效位置 1。请注意决不要预先加载或改变 TMR1L 寄存器，这样做可能会引起多个周期的累积错误。

要使此方法够精确，如程序 `RTCinit` 所示，Timer1 必须工作在异步模式且必须使能 Timer1 溢出中断（PIE1<0> = `1`）。同时 Timer1 振荡器也必须使能并始终保持运行。

 超前信息

例 **12-1:** 使用 **TIMER1** 中断服务实现实时时钟

```
RTCinit
         MOVLW     0x80                      ; Preload TMR1 register pair
         MOVWF     TMR1H                     ; for 1 second overflow
         CLRF      TMR1L
         MOVLW     b’00001111’               ; Configure for external clock,
         MOVWF     T1OSC                     ; Asynchronous operation, external oscillator
         CLRF      secs                      ; Initialize timekeeping registers
         CLRF      mins                      ;
         MOVLW     .12
         MOVWF     hours
         BSF       PIE1, TMR1IE              ; Enable Timer1 interrupt
         RETURN
RTCisr
         BSF       TMR1H, 7                  ; Preload for 1 sec overflow
         BCF       PIR1, TMR1IF              ; Clear interrupt flag
         INCF      secs, F                   ; Increment seconds
         MOVLW     .59                       ; 60 seconds elapsed?
         CPFSGT    secs
         RETURN                              ; No, done
         CLRF      secs                      ; Clear seconds
         INCF      mins, F                   ; Increment minutes
         MOVLW     .59                       ; 60 minutes elapsed?
         CPFSGT    mins
         RETURN                              ; No, done
         CLRF      mins                      ; clear minutes
         INCF      hours, F                  ; Increment hours
         MOVLW     .23                       ; 24 hours elapsed?
         CPFSGT    hours
         RETURN                              ; No, done
         MOVLW     .01                       ; Reset hours to 1
         MOVWF     hours
         RETURN                              ; Done
```

表 **12-2:** **TIMER1** 作为定时器 **/** 计数器时相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
TMR1L	Timer1 寄存器的低字节								42
TMR1H	Timer1 寄存器的高字节								42
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	42

图注： x = 未知，u = 不变，— = 未用 （读为 0）。
Timer1 模块不使用阴影单元。

注：

13.0 电源控制 PWM 模块

在电机控制和电源转换应用中，电源控制 PWM 模块简化了产生多个同步脉宽调制（PWM）输出信号的工作。PWM 模块尤其支持以下电源和电机控制应用：

- 三相和单相交流感应电机
- 开关磁阻电机
- 无刷直流 （Brushless DC， BLDC）电机
- 不间断电源 （Uninterruptible Power Supplies，UPS）
- 多种有刷直流电机

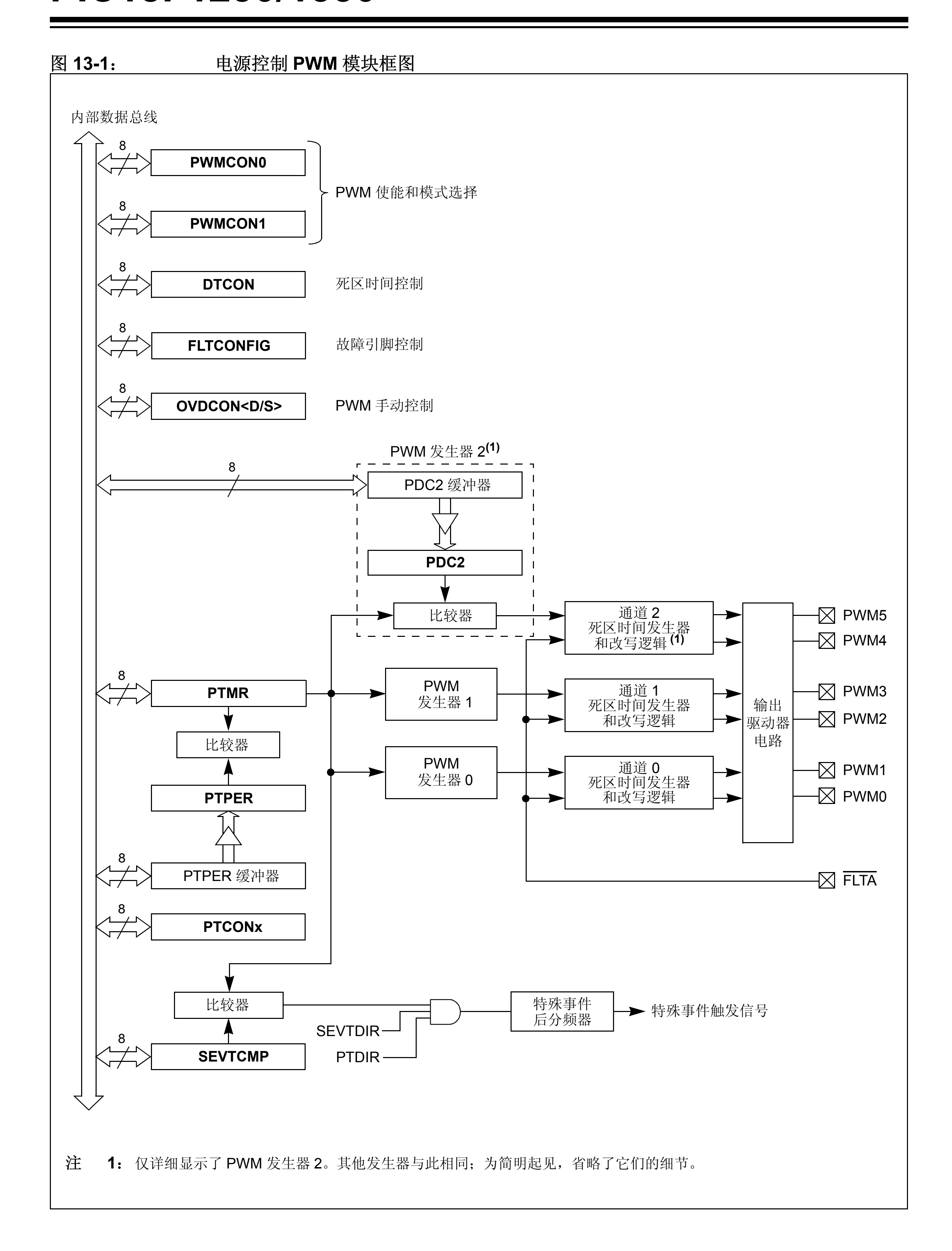
PWM 模块具有以下特性：

- 多达 6 个 PWM I/O 引脚，带有 3 个占空比发生器。可将引脚两两组对以实现完整的半桥控制。
- 取决于 PWM 周期，分辨率最高可达 14 位 。
- 可在工作过程中随时改变 PWM 的频率。
- 边沿和中心对齐输出模式。
- 单脉冲发生模式。
- 可通过编程对成对的 PWM 信号之间的死区时间进行控制。
- 支持中心对齐模式下的非对称更新中断。
- 例如 BLDC 等电子换向电机 （Electrically Commutated Motor， ECM）的输出改写控制。
- 用于触发 A/D 转换的特殊事件触发比较器。
- 在调试模式下， PWM 输出禁止功能可将 PWM 输出设置为无效状态。

PIC18F1230/1330 器件的电源控制 PWM 模块支持 3 个 PWM 发生器和 6 路输出通道。图 13-1 给出了该模块的简化框图。图 13-2 和图 13-3 分别给出了互补和独立输出模式下，每对 PWM 输出的硬件配置方式。

后面的小节将针对 PWM 模块的每个功能单元进行讨论。

图 13-1： 电源控制 PWM 模块框图



注 1：仅详细显示了 PWM 发生器 2。其他发生器与此相同；为简明起见，省略了它们的细节。

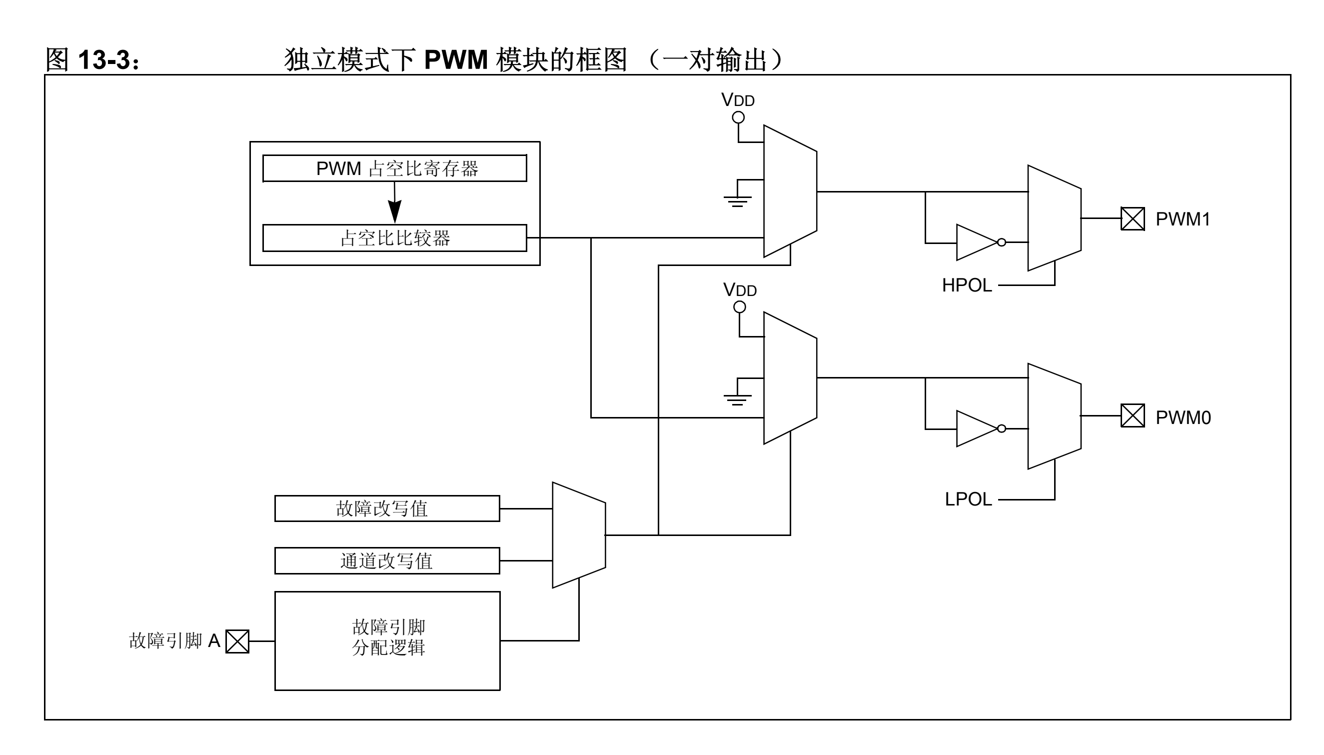
 超前信息

图 **13-2**： 互补模式下 **PWM** 模块的框图 （一对输出）

VDD
死区
发生器
PWM1
占空比比较器
HPOL
PWM 占空比寄存器
PWM0
LPOL
故障改写值
通道改写值
故障引脚
分配逻辑
故障引脚 A

注： 在互补模式下，如果奇数通道有效，那么偶数通道将无法被故障或改写事件强制有效。偶数通道的输出始终与奇数通道输出互补并保持无效，当奇数通道被驱动为其有效状态之前，会插入一段死区时间。

图 **13-3**： 独立模式下 **PWM** 模块的框图 （一对输出）



该模块包含三个占空比发生器，编号从 0 至 2 ；六个 PWM 输出引脚，编号从 0 至 5。这六路 PWM 输出两两组对，每对均具有一个奇数编号的输出和一个偶数编号的输出。在互补模式下，偶数 PWM 引脚的输出必须始终与相应奇数 PWM 引脚输出互补。例如，PWM0 与 PWM1 互补，而 PWM2 与 PWM3 互补。在引脚对中的某个引脚变为无效（OFF）和与其互补的引脚变为有效（ON）前，死区时间发生器会插入一段 OFF 延迟 （称为 “死区时间”）。这样做的目的是为了保护与 PWM 输出引脚相连的功率开关设备免遭破坏。

PWM 模块的时基是由其自带的 12 位定时器提供的，也可选择将该定时器与预分频器和后分频器配合使用。

13.1 控制寄存器

共有 20 个寄存器用于控制 PWM 模块的操作。其中 8 个用于配置该模块的功能：

- PWM 定时器控制寄存器 0 （PTCON0）
- PWM 定时器控制寄存器 1 （PTCON1）
- PWM 控制寄存器 0 （PWMCON0）
- PWM 控制寄存器 1 （PWMCON1）
- 死区时间控制寄存器 （DTCON）
- 输出改写控制寄存器 （OVDCOND）
- 输出状态寄存器 （OVDCONS）
- 故障配置寄存器 （FLTCONFIG）

还有 12 个寄存器被配置为 6 组 16 位寄存器 。它们用于特定功能的配置值。它们是：

- PWM 时基寄存器 （PTMRH 和 PTMRL）
- PWM 时基周期寄存器 （PTPERH 和 PTPERL）
- PWM 特殊事件比较寄存器 （SEVTCMPH 和 SEVTCMPL）
- PWM 占空比寄存器 0 （PDC0H 和 PDC0L）
- PWM 占空比寄存器 1 （PDC1H 和 PDC1L）
- PWM 占空比寄存器 2 （PDC2H 和 PDC2L）

所有这些寄存器都是双重缓冲的。

13.2 模块的功能

PWM 模块支持几种对特定电源和电机控制应用有利的工作模式。后面的小节将一一描述每种工作模式。

PWM 模块由几个功能模块组成。每个功能模块的操作将在相应的工作模式中分别介绍：

- PWM 时基
- PWM 时基中断
- PWM 周期
- PWM 占空比
- 死区时间发生器
- PWM 输出改写逻辑
- PWM 故障输入引脚
- PWM 特殊事件触发器

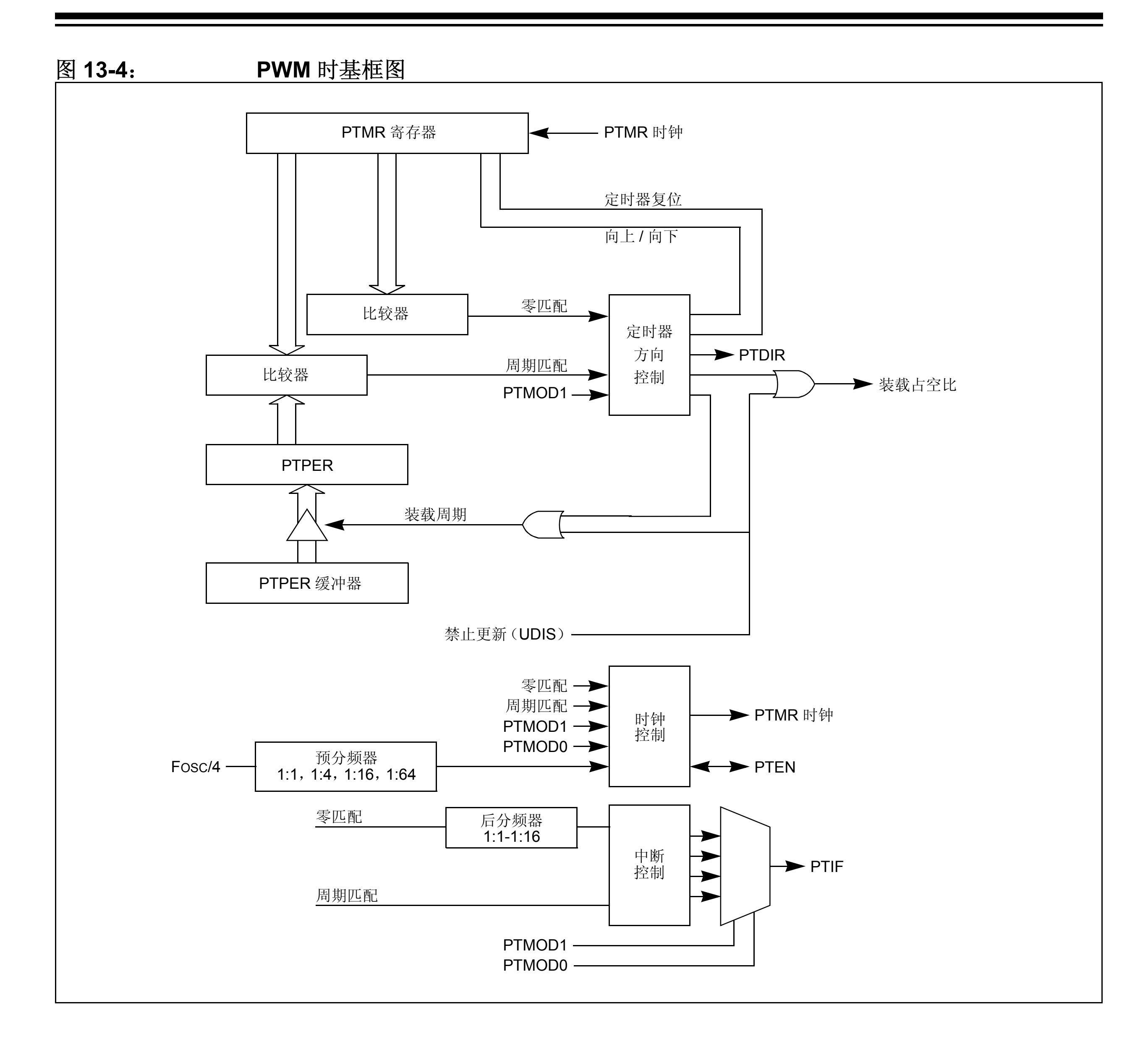
13.3 PWM 时基

PWM 时基由带有预分频和后分频功能的 12 位定时器提供。图 13-4 给出 了 PWM 时基的简化框图。由 PTCON0 和 PTCON1 寄存器配置 PWM 时基。通过将 PTCON1 寄存器中的 PTEN 位分别置 1 或清零可使能或禁止该时基。

注：	当在软件中清零 PTEN 位时，不会清零 PTMR 寄存器对 （PTMRL:PTMRH）。

 超前信息

图 13-4：　　PWM 时基框图



可将 PWM 时基配置为 4 种工作模式：

- 自由运行模式
- 单次触发模式
- 连续向上 / 向下计数模式
- 在两次更新时均产生中断的连续向上 / 向下计数模式

由 PTCON0 寄存器中的 PTMOD1:PTMOD0 位选择这 4 种模式中的一种。自由运行模式产生边沿对齐的 PWM 信号。连续向上 / 向下计数模式产生中心对齐的 PWM 信号。单次触发模式允许 PWM 模块使用边沿对齐的脉冲对某些电子换向电机（ECM）进行控制。

寄存器 13-1： PTCON0：PWM 定时器控制寄存器 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-4 **PTOPS3:PTOPS0**：PWM 时基输出后分频比选择位
0000 = 1:1 后分频比
0001 = 1:2 后分频比
.
.
.
1111 = 1:16 后分频比

bit 3-2 **PTCKPS1:PTCKPS0**：PWM 时基输入时钟预分频比选择位
00 = PWM 时基输入时钟的频率为 Fosc/4 （1:1 预分频比）
01 = PWM 时基输入时钟的频率为 Fosc/16 （1:4 预分频比）
10 = PWM 时基输入时钟的频率为 Fosc/64 （1:16 预分频比）
11 = PWM 时基输入时钟的频率为 Fosc/256 （1:64 预分频比）

bit 1-0 **PTMOD1:PTMOD0**：PWM 时基模式选择位
11 = PWM 时基工作在两次更新时均产生中断的连续向上 / 向下计数模式
10 = PWM 时基工作在连续向上 / 向下计数模式
01 = PWM 时基工作在单次触发模式
00 = PWM 时基工作在自由运行模式

寄存器 13-2： PTCON1：PWM 定时器控制寄存器 1

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
PTEN	PTDIR	—	—	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **PTEN**：PWM 时基定时器使能位
1 = 使能 PWM 时基
0 = 禁止 PWM 时基

bit 6 **PTDIR**：PWM 时基计数方向状态位
1 = PWM 时基向下计数
0 = PWM 时基向上计数

bit 5-0 未用：读为 0

寄存器 13-3： PWMCON0：PWM 控制寄存器 0

U-0	R/W-1[1]	R/W-1[1]	R/W-1[1]	U-0	R/W-0	R/W-0	R/W-0
—	PWMEN2	PWMEN1	PWMEN0	—	PMOD2	PMOD1	PMOD0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7　未用：读为 0

bit 6-4　**PWMEN2:PWMEN0**：PWM 模块使能位 [1]

111 = 所有编号为奇数的 PWM I/O 引脚被使能为 PWM 输出
110 = PWM1 和 PWM3 引脚被使能为 PWM 输出
10x = 所有 PWM I/O 引脚被使能为 PWM 输出
011 = PWM0、 PWM1、 PWM2 和 PWM3 I/O 引脚被使能为 PWM 输出
010 = PWM0 和 PWM1 引脚被使能为 PWM 输出
001 = PWM1 引脚被使能为 PWM 输出
000 = 禁止 PWM 模块；所有 PWM I/O 引脚用作通用 I/O

bit 3　未用：读为 0

bit 2-0　**PMOD2:PMOD0**：PWM 输出对模式位

对于 PMOD0：
1 = PWM I/O 引脚对 （PWM0 和 PWM1）处于独立模式
0 = PWM I/O 引脚对 （PWM0 和 PWM1）处于互补模式

对于 PMOD1：
1 = PWM I/O 引脚对 （PWM2 和 PWM3）处于独立模式
0 = PWM I/O 引脚对 （PWM2 和 PWM3）处于互补模式

对于 PMOD2：
1 = PWM I/O 引脚对 （PWM4 和 PWM5）处于独立模式
0 = PWM I/O 引脚对 （PWM4 和 PWM5）处于互补模式

注 **1**： PWMEN 位的复位状态取决于 PWMPIN 配置位的值。

寄存器 13-4： PWMCON1：PWM 控制寄存器 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-4 **SEVOPS3:SEVOPS0**：PWM 特殊事件触发输出信号后分频比选择位
`0000` = 1:1 后分频比
`0001` = 1:2 后分频比
.
.
.
`1111` = 1:16 后分频比

bit 3 **SEVTDIR**：特殊事件触发信号时基方向位
`1` = 当 PWM 时基向下计数时产生特殊事件触发信号
`0` = 当 PWM 时基向上计数时产生特殊事件触发信号

bit 2 未用：读为 `0`

bit 1 **UDIS**：PWM 更新使能位
`1` = 禁止来自占空比寄存器和周期缓冲器的更新
`0` = 允许来自占空比寄存器和周期缓冲器的更新

bit 0 **OSYNC**：PWM 输出改写同步位
`1` = 通过 OVDCON 寄存器改写输出的操作与 PWM 时基同步
`0` = 通过 OVDCON 寄存器改写输出的操作与 PWM 时基异步

 超前信息

13.3.1 自由运行模式

在自由运行模式下，PWM 时基（PTMRL 和 PTMRH）向上计数直到其值与 PWM 时基周期寄存器 PTPER（PTPERL 和 PTPERH）中的值匹配。在输入时钟的下一个边沿，PTMR 寄存器复位，并且只要 PTEN 位保持置 1，时基就将继续向上计数。

13.3.2 单次触发模式

在单次触发模式下，当 PTEN 置 1 时， PWM 时基就将开始向上计数。当 PTMR 寄存器中的值与 PTPER 寄存器中的值匹配时，PTMR 寄存器就会在输入时钟的下一个边沿复位，并且 PTEN 会被硬件清零，从而时基暂停计数。

13.3.3 连续向上 / 向下计数模式

在连续向上 / 向下计数模式下， PWM 时基向上计数直到 PTPER 寄存器的值与 PTMR 寄存器的值匹配。在输入时钟的下一个边沿，该定时器向下计数。PTCON1 寄存器中的 PTDIR 位是只读的，用于表示计数的方向。当定时器向下计数时，该位置 1。

注：	由于PWM比较输出会在PWM时基开始向下计数且与占空比值匹配时被驱动为有效状态，所以在连续向上 / 向下计数模式第一个周期的前半部分，PWM 输出始终保持无效，直到 PTMR 从 PTPER 值开始向下计数时， PWM 比较输出才被驱动为有效状态。

13.3.4 PWM 时基预分频器

输入给 PTMR 的时钟 （频率为 FOSC/4）具有以下几种预分频比选项：1:1、 1:4、 1:16 或 1:64。预分频比由 PTCON0 寄存器中的控制位 PTCKPS<1:0> 选择。当以下任一事件发生时，预分频计数器清零：

- 写入 PTMR 寄存器
- 写入 PTCON （PTCON0 或 PTCON1）寄存器
- 任何器件复位

注：	当写入 PTCONx 寄存器时，PTMR 寄存器不会清零。

表 13-1 给出了可由 PWM 时基和预分频器产生的最小 PWM频率。此表假设工作频率为40 MHz（FCYC = 10 MHz）且 PTPER = 0xFFF。 PWM 模块必须能够产生输电线频率 （50 Hz 或 60 Hz）的 PWM 信号，以适应某些电源控制应用的需求。

表 13-1： 最小 PWM 频率

最小 PWM 频率与预分频比的关系（ FCYC = 10 MIPS， PTPER = 0FFFh）		
预分频比	**PWM 频率**（边沿对齐）	**PWM 频率**（中心对齐）
1:1	2441 Hz	1221 Hz
1:4	610 Hz	305 Hz
1:16	153 Hz	76 Hz
1:64	38 Hz	19 Hz

13.3.5 PWM 时基后分频器

可以选择将PTMR的匹配输出信号通过一个4位的后分频器 （可提供 1:1 至 1:16 分频比）进行后分频，以产生一个中断。当以下任一事件发生时，后分频计数器清零：

- 写入 PTMR 寄存器
- 写入 PTCONx 寄存器
- 任何器件复位

当写入 PTCONx 寄存器时， PTMR 寄存器不会清零。

13.4 PWM 时基中断

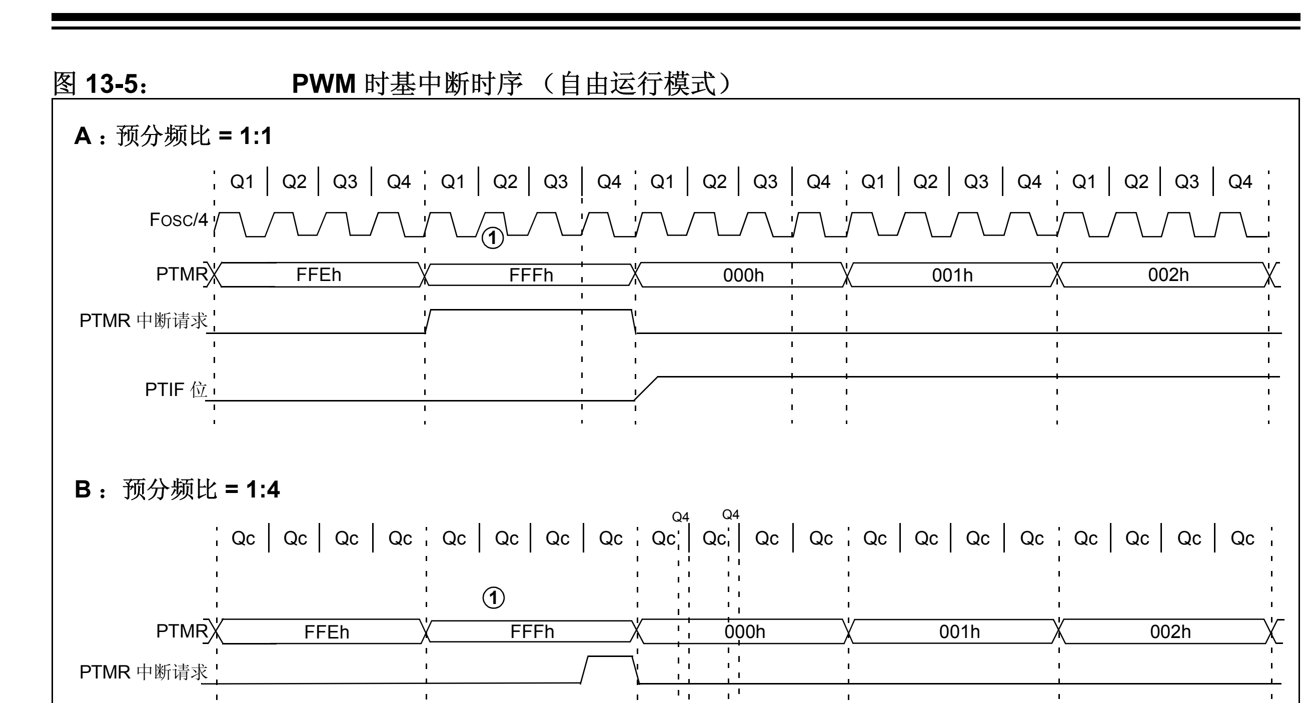
PWM定时器可根据 PTMOD<1:0> 选择的工作模式和后分频比选择位 （PTOPS<3:0>）的值产生中断。

13.4.1 自由运行模式下的中断

当 PWM 时基工作在自由运行模式 （PTMOD<1:0> = 00）时，每当其值与 PTPER 寄存器中的值匹配时，就会产生中断。在时钟的下一个边沿，PTMR 寄存器复位为零。

使用除 1:1 外的后分频比将降低中断事件发生的频率。

图 13-5： PWM 时基中断时序 （自由运行模式）



注 1：本例中， PWM 时基周期寄存器 PTPER 装载的值是 0FFFh。

13.4.2 单次触发模式下的中断

当 PWM 时基工作在单次触发模式（PTMOD<1:0> = 01）时，若其值与 PTPER 寄存器匹配，则会产生一个中断。 PWM 时基寄存器 （PTMR）在输入时钟的下一个边沿复位为零，并且 PTEN 位清零。后分频器选择位对该定时器模式没有影响。

13.4.3 连续向上 / 向下计数模式下的中断

在连续向上 / 向下计数模式（PTMOD<1:0> = 10）中，每当 PTMR 计数器为零且 PWM 时基开始向上计数时，就会产生中断。可以在该定时器模式中使用后分频比选择位以降低中断事件发生的频率。图 13-7 显示了连续向上 / 向下计数模式中的中断。

 超前信息

图 13-6： PWM 时基中断时序 （单次触发模式）

A：预分频比 = 1:1

Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4

FOSC/4

PTMR FFEh FFFh 000h 000h 000h

PTMR 中断请求

PTIF 位

B：预分频比 = 1:4

Q4 Q4

Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc | Qc

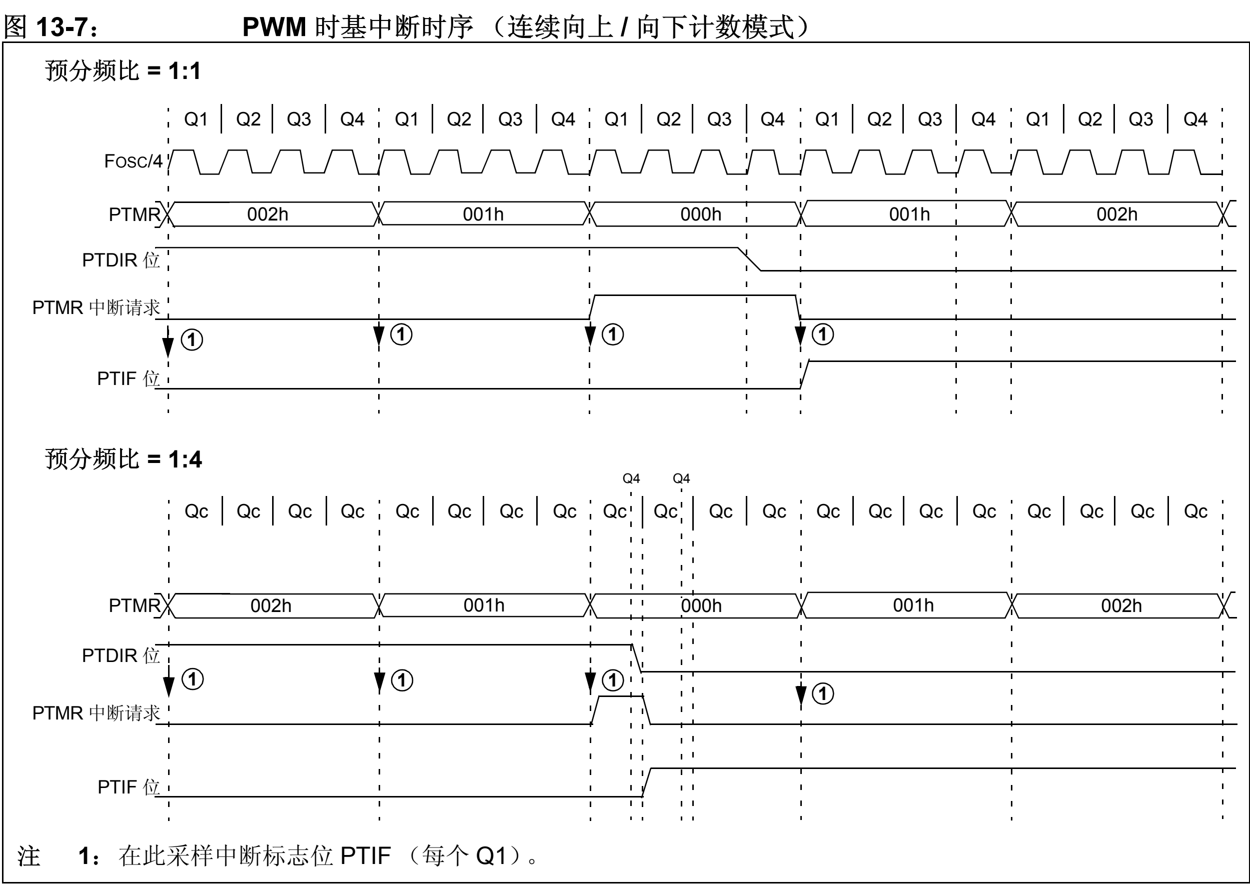
PTMR FFEh FFFh 000h 000h 000h

PTMR 中断请求

PTIF 位

注 1：在此采样中断标志位 PTIF （每个 Q1）。
2：本例中， PWM 时基周期寄存器 PTPER 装载的值是 0FFFh。

图 13-7： PWM 时基中断时序 （连续向上 / 向下计数模式）



注 1：在此采样中断标志位 PTIF （每个 Q1）。

13.4.4 双更新模式下的中断

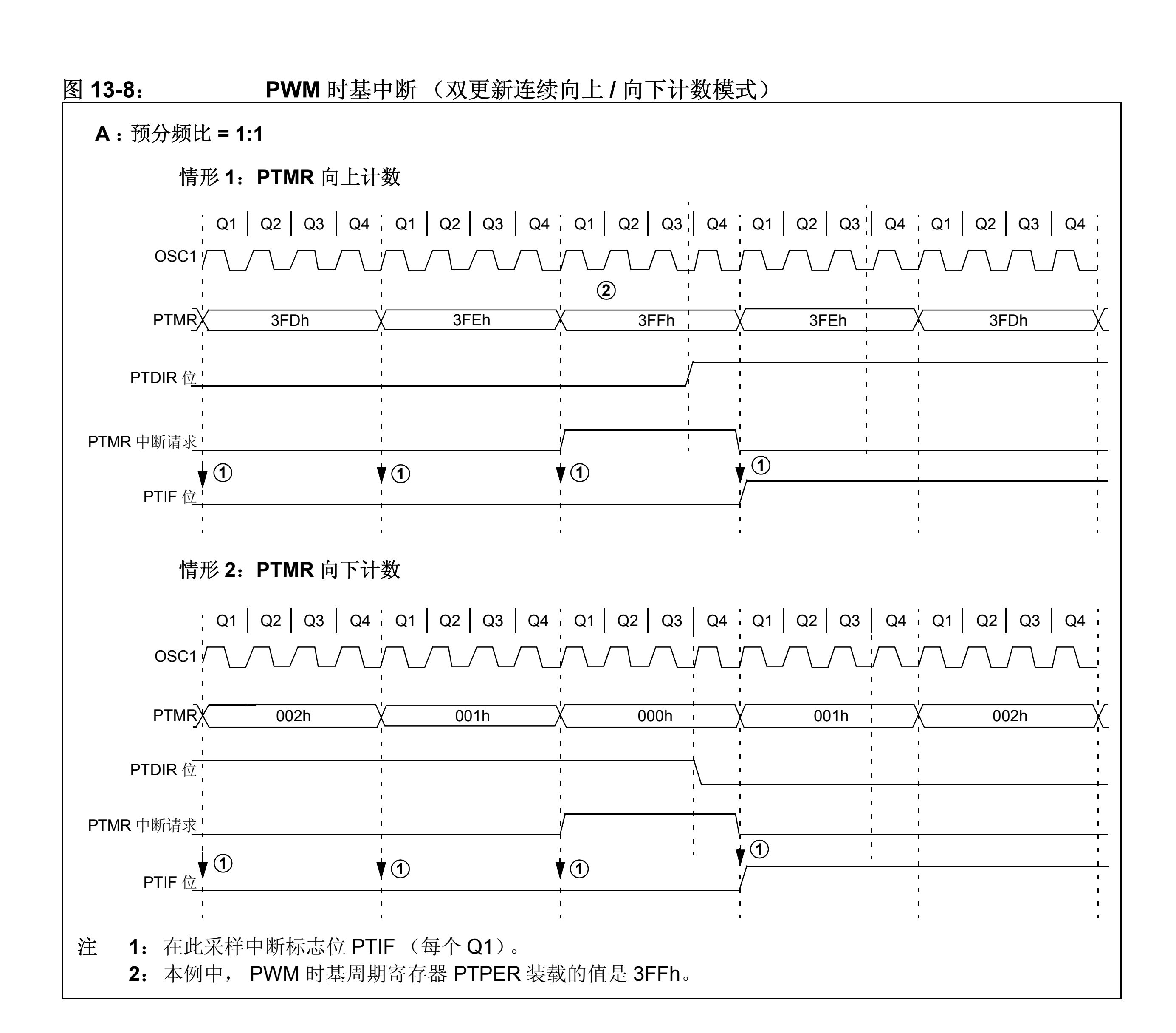
此模式只可用在连续向上 / 向下计数模式中。在双更新模式 （PTMOD<1:0> = 11）中，每当 PTMR 寄存器等于零或 PTMR 与 PTPER 寄存器匹配时，就会产生中断。图13-8 给出了双更新连续向上/向下计数模式中的中断。

双更新模式在中心对齐模式下向用户提供了两种额外的功能。

1. 由于 PWM 占空比可在每个周期更新两次，因而使控制环带宽加倍。
2. 可产生非对称的中心对齐 PWM 波形，这种波形有助于在特定电机控制应用中使输出波形失真减至最小。

注：	当 PTEN 位有效时，不要改变 PTMOD 位的值，否则将导致不可预料的结果。要改变 PWM 定时器的工作模式，请首先将 PTEN 位清零，并使用所需的数据装载 PTMOD 位，然后重新将 PTEN 位置 1。

图 13-8： PWM 时基中断 （双更新连续向上 / 向下计数模式）



 超前信息

13.5 PWM 周期

PWM 周期是由 PTPER 寄存器对（PTPERL 和 PTPERH）定义的。 PTPERH 的低 4 位和 PTPERL 中的 8 位构成的 PWM 周期具有 12 位分辨率。PTPER 是一个双重缓冲的寄存器，用于设置 PWM 时基的计数周期。

在以下时刻，PTPER 缓冲器的内容被装入 PTPER 寄存器：

- 自由运行和单次触发模式：当 PTMR 的值与 PTPER 的值匹配后 PTMR 寄存器复位为零时。
- 连续向上 / 向下计数模式：当 PTMR 寄存器的值为零时。当禁止 PWM 时基 （PTEN = 0）时， PTPER 缓冲器中的值会自动装入 PTPER 寄存器。图 13-9 和图 13-10 指出了 PTPER 缓冲器中的内容被装入实际的 PTPER 寄存器的时刻。

可使用以下公式计算 PWM 周期：

公式 13-1： 自由运行模式下的 PWM 周期

$$T_{PWM} = \frac{(PTPER + 1) \times PTMRPS}{F_{OSC}/4}$$

公式 13-2： 连续向上 / 向下计数模式下的 PWM 周期

$$T_{PWM} = \frac{(2 \times PTPER) \times PTMRPS}{\frac{F_{OSC}}{4}}$$

PWM 频率是 PWM 周期的倒数，如下：

公式 13-3： PWM 频率

$$\text{PWM 频率} = \frac{1}{\text{PWM 周期}}$$

在给定器件振荡器和 PWM 频率的情况下，最大分辨率（单位为位）可由以下公式确定：

公式 13-4： PWM 分辨率

$$\text{分辨率} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)}$$

表13-2给出了选择不同执行速率和PTPER值时的 PWM 分辨率和频率。表 13-2 中的 PWM 频率是在边沿对齐的 PWM 模式下计算的。对于中心对齐模式，PWM 频率大约是表中所示频率的一半。

表 13-2： PWM 频率和分辨率示例

PWM 频率 = 1/TPWM

Fosc	MIPS	PTPER 值	PWM 分辨率	PWM 频率
40 MHz	10	0FFFh	14 位	2.4 kHz
40 MHz	10	07FFh	13 位	4.9 kHz
40 MHz	10	03FFh	12 位	9.8 kHz
40 MHz	10	01FFh	11 位	19.5 kHz
40 MHz	10	FFh	10 位	39.0 kHz
40 MHz	10	7Fh	9 位	78.1 kHz
40 MHz	10	3Fh	8 位	156.2 kHz
40 MHz	10	1Fh	7 位	312.5 kHz
40 MHz	10	0Fh	6 位	625 kHz
25 MHz	6.25	0FFFh	14 位	1.5 kHz
25 MHz	6.25	03FFh	12 位	6.1 kHz
25 MHz	6.25	FFh	10 位	24.4 kHz
10 MHz	2.5	0FFFh	14 位	610 Hz
10 MHz	2.5	03FFh	12 位	2.4 kHz
10 MHz	2.5	FFh	10 位	9.8 kHz
5 MHz	1.25	0FFFh	14 位	305 Hz
5 MHz	1.25	03FFh	12 位	1.2 kHz
5 MHz	1.25	FFh	10 位	4.9 kHz
4 MHz	1	0FFFh	14 位	244 Hz
4 MHz	1	03FFh	12 位	976 Hz
4 MHz	1	FFh	10 位	3.9 kHz

注： 对于中心对齐模式，PWM 频率大约是表中所示频率的一半。

图 13-9： 在自由运行模式下更新 PWM 周期缓冲器

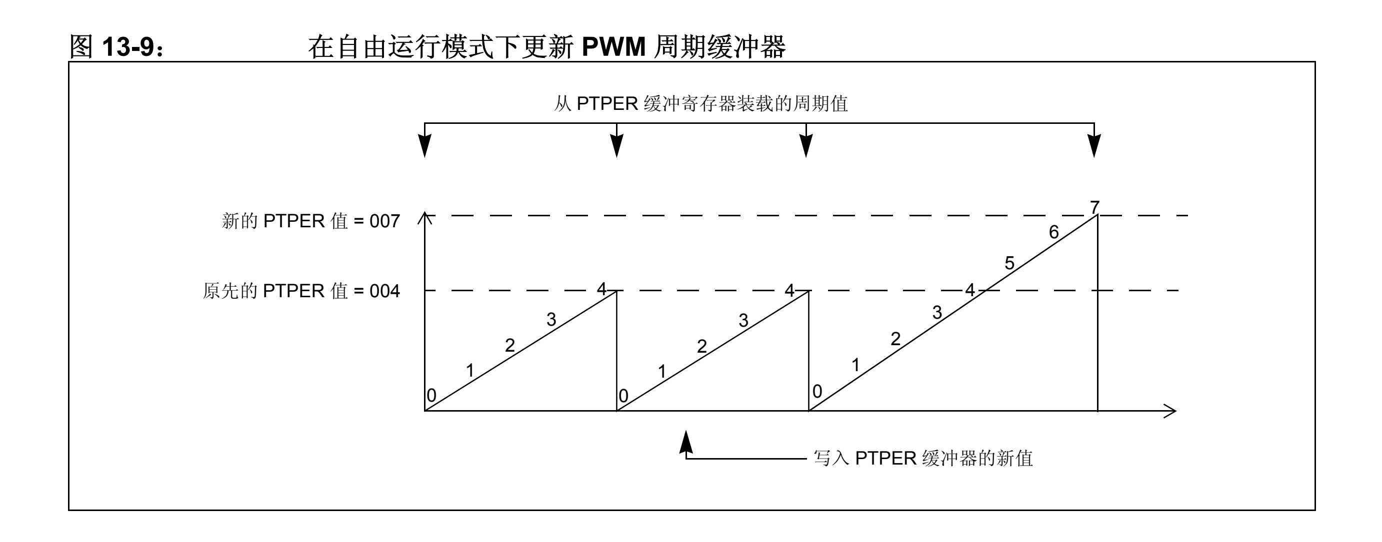
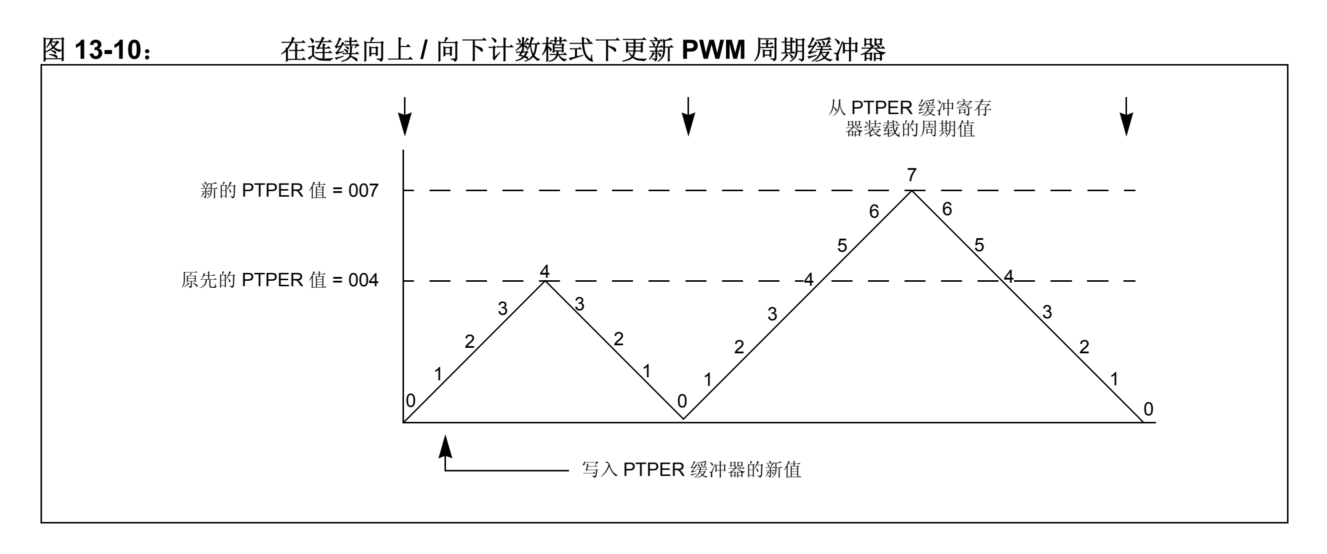


图 13-10： 在连续向上 / 向下计数模式下更新 PWM 周期缓冲器



 超前信息

13.6 PWM 占空比

PWM 占空比是由 PDCx （PDCxL 和 PDCxH）寄存器定义的。总共有三个PWM占空比寄存器用于四对PWM 通道。PDCxH 的低 6 位与 PDCxL 的 8 位组成的占空比寄存器具有 14 位分辨率。PDCx 是一个双重缓冲的寄存器，用于设置 PWM 时基的计数周期。

13.6.1 PWM 占空比寄存器

有 3 个 14 位特殊功能寄存器用于指定 PWM 模块的占空比值：

- PDC0 （PDC0L 和 PDC0H）
- PDC1 （PDC1L 和 PDC1H）
- PDC2 （PDC2L 和 PDC2H）

每个占空比寄存器中的值决定 PWM 输出保持在有效状态的时间。PDCx 的高 12 位保存 PTMRH/L<11:0> 中实际的占空比值，而低 2 位控制占空比匹配发生在哪个内部 Q 时钟上。如图 13-11 所示，当预分频比为 1:1 （PTCKPS<1:0> = 00）时，此 2 位值被译码为相应的 Q 时钟。

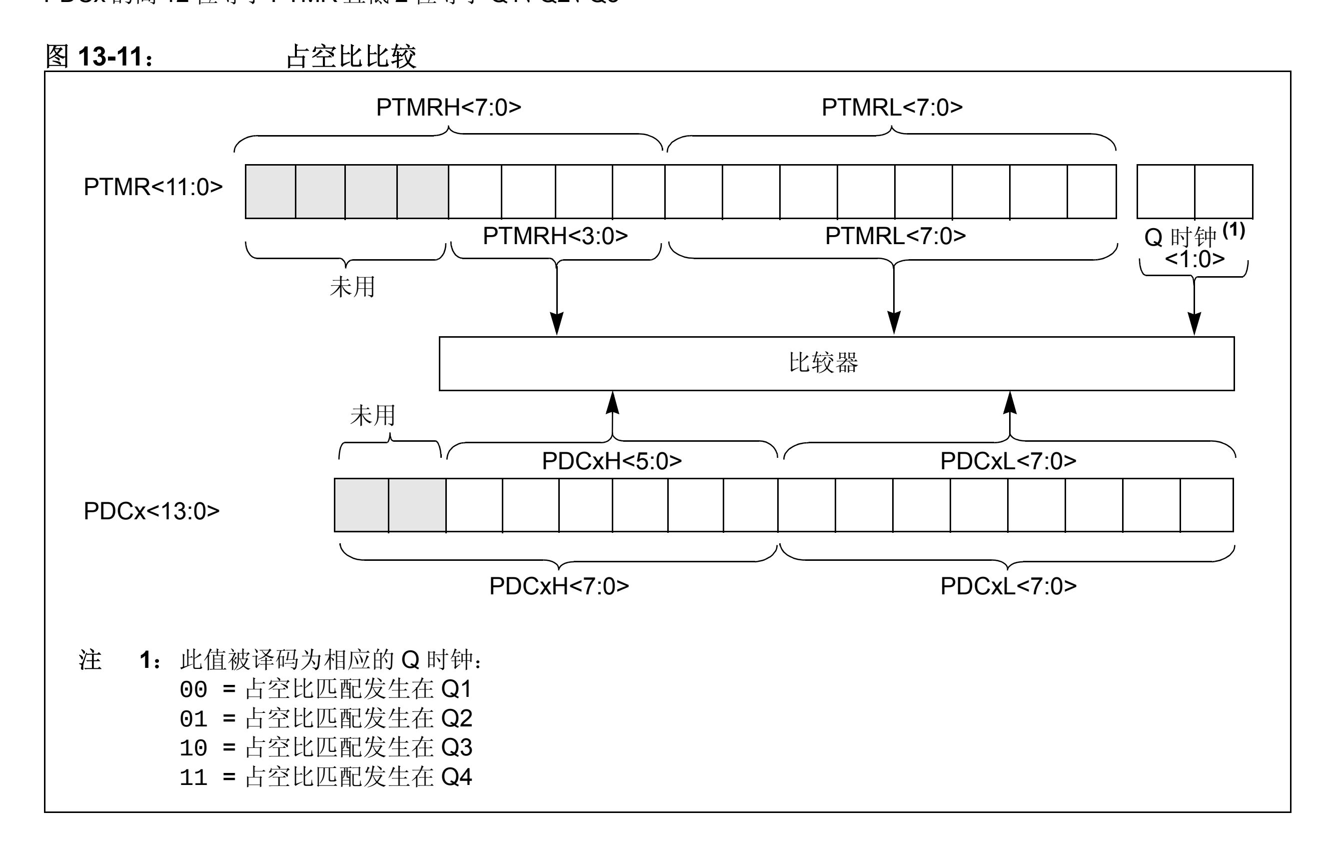
在边沿对齐模式下，PWM 周期在 Q1 开始，当占空比寄存器与 PTMR 寄存器匹配时 （如下文所述）结束。当 PDCx 的高 12 位等于 PTMR 且低 2 位等于 Q1、Q2、Q3 或 Q4 时认为发生了占空比匹配，具体是哪个 Q 节拍由 PDCx 的低两位 （当预分频比为 1:1 或 PTCKPS <1:0> = 00 时）决定。

注：	当预分频比不为 1:1（PTCKPS<1:0> ≠ 00）时，当 PTMR 与 PDCx 匹配时，占空比匹配发生在指令周期的 Q1 时钟上。

每个比较单元都有允许改写 PWM 信号的逻辑。此逻辑还能确保 PWM 信号在互补模式中是彼此互补的（改变状态时会插入死区时间）（见第 **13.7 节 “死区时间发生器 ”**）。

注：	为了获得正确的 PWM 占空比，在装载 PWM 占空比寄存器之前，要先将计算得到的 PWM 占空比值乘以 4。这是由于 PWM 占空比寄存器的两个额外 LSB 要与内部 Q 时钟比较以确定发生PWM占空比匹配的时间。

图 13-11： 占空比比较



13.6.2 占空比缓冲寄存器

3 个 PWM 占空比寄存器是双重缓冲的，以避免在更新 PWM 输出时产生毛刺。对于每个占空比电路来说，都有一个可由用户访问的占空比缓冲寄存器和一个用于保存当前 PWM 周期实际使用的比较值的辅助占空比寄存器。

如图 13-12 所示，在边沿对齐的 PWM 输出模式中，只要 PTMR 与 PTPER 寄存器发生匹配并且 PTMR 复位，新的占空比值就会被更新。此外，当禁止 PWM 时基（PTEN = 0）时，占空比缓冲器的内容会自动装入占空比寄存器。

如果 PWM 时基处于连续向上 / 向下计数模式，当 PTMR 寄存器的值为零并且 PWM 时基开始向上计数时，新占空比将被更新。当禁止 PWM 时基（PTEN = 0）时，占空比缓冲器的内容被自动装入占空比寄存器。图 13-13 给出了连续向上 / 向下计数器模式中发生占空比更新的时序。在此模式中，在更改生效之前，最长会有一个 PWM 周期的时间用于计算和装入新的 PWM 占空比。

当 PWM 时基处于双更新连续向上 / 向下计数模式时，如果 PTMR 寄存器的值为零或者 PTMR 寄存器的值与 PTPER 寄存器中的值匹配时，新的占空比值将被更新。在上述两个条件发生时，占空比缓冲器的内容被自动装入占空比寄存器。图 13-4 给出了双更新连续向上 / 向下计数模式的占空比更新的时序。在此模式中，在更改生效之前，最长会有半个 PWM 周期的时间用于计算和装入新的 PWM 占空比。

13.6.3 边沿对齐的 PWM

当 PWM 时基工作在自由运行模式或单次触发模式时，模块产生边沿对齐的PWM信号。对于边沿对齐的PWM输出，给定PWM通道输出信号的周期由PTPER中装载的值指定，其占空比则由相应占空比寄存器指定（见图 13-12）。在周期开始时（PTMR = 0），PWM 输出被驱动为有效；而当占空比寄存器中的值与 PTMR 匹配时，PWM 输出被驱动为无效。如 PWM 周期章节所述，当 PTMR 与 PTPER 匹配时，新的周期开始。

如果特定占空比寄存器中的值为 0，则相应 PWM 引脚的输出在整个 PWM 周期中都将无效。此外，如果占空比寄存器的值大于 PTPER 寄存器中保存的值，那么 PWM 引脚上的输出将在整个 PWM 周期有效。

图 13-12： 边沿对齐的 PWM

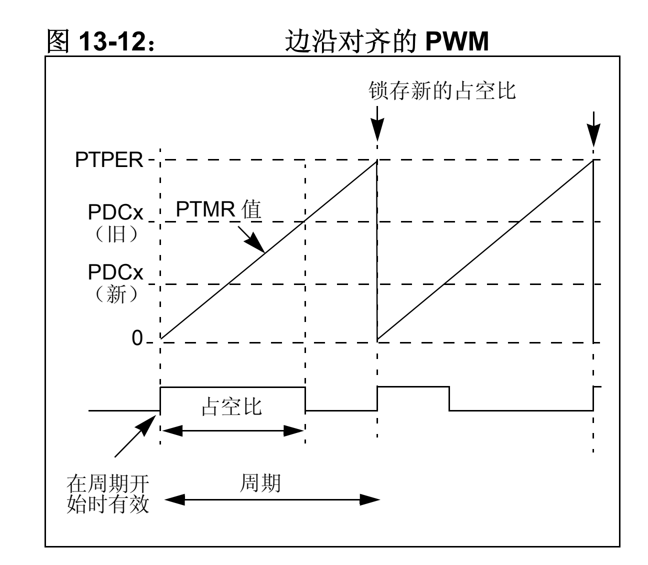
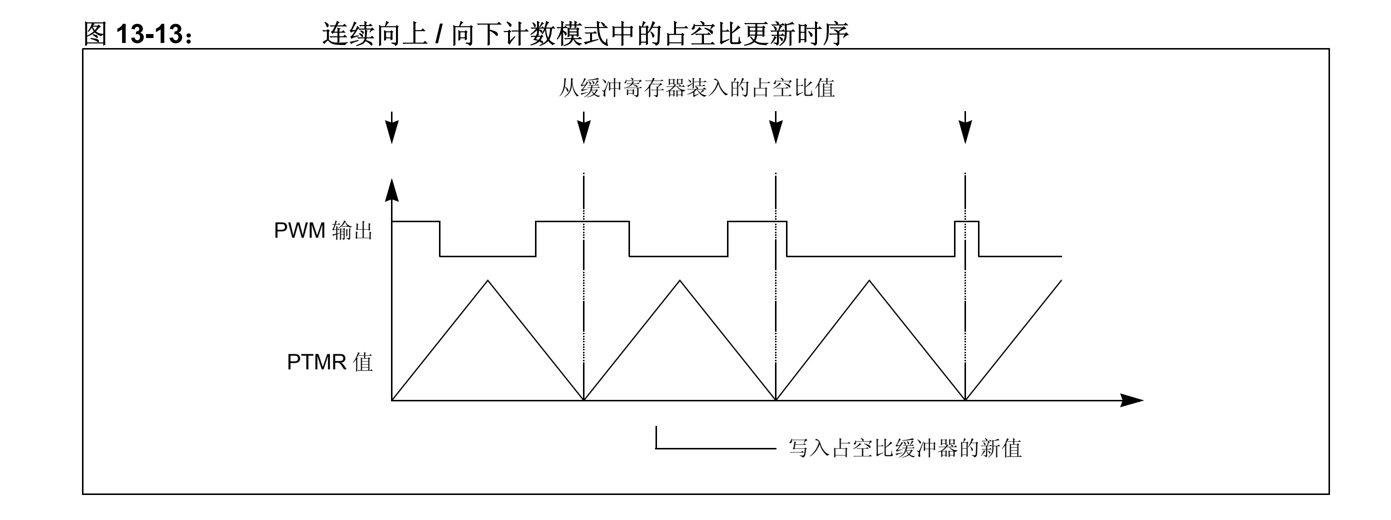
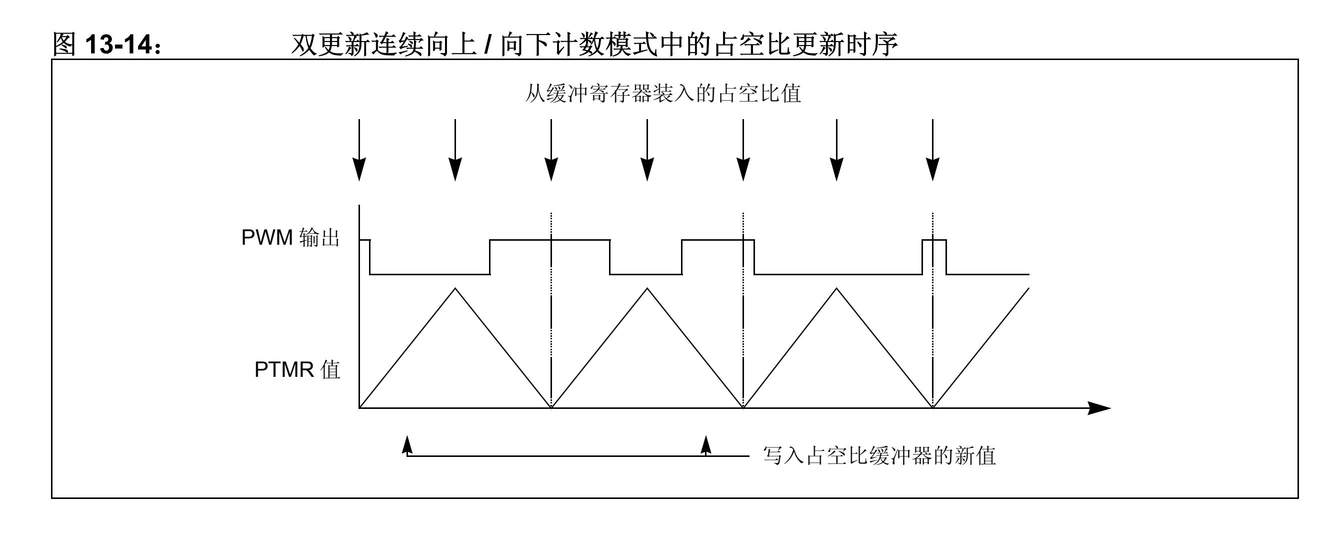


图 13-13： 连续向上 / 向下计数模式中的占空比更新时序



 超前信息

图 13-14： 双更新连续向上 / 向下计数模式中的占空比更新时序

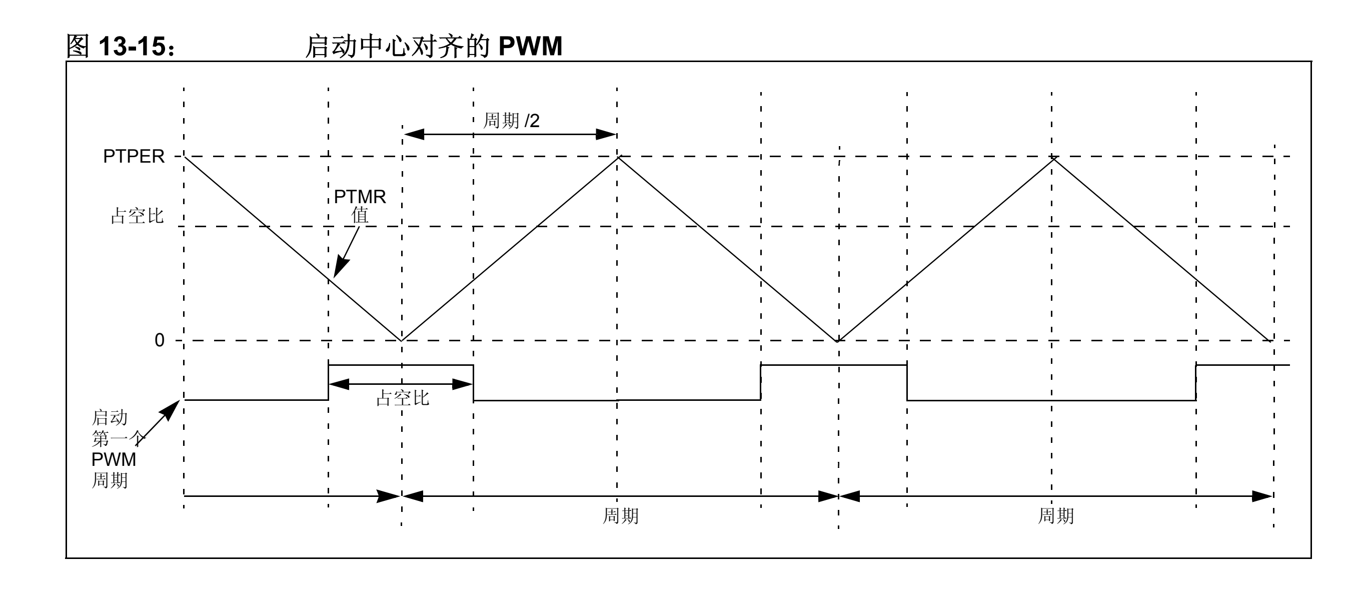


13.6.4 中心对齐的 PWM

当将 PWM 时基配置为工作在连续向上 / 向下计数模式时，模块产生中心对齐的 PWM 信号（见图 13-15）。当占空比寄存器的值与PTMR 的值匹配且PWM 时基向下计数（PTDIR = 1）时，PWM 比较输出被驱动为有效状态。当 PWM 时基开始向上计数（PTDIR = 0）且 PTMR 寄存器中的值与占空比值匹配时，PWM 比较输出将被驱动为无效状态。如果特定占空比寄存器中的值为 0，则相应 PWM 引脚的输出在整个 PWM 周期中都为无效。此外，如果占空比寄存器的值大于或等于 PTPER寄存器的值，PWM引脚上的输出将在整个PWM 周期内有效。

> **注：** 当 PWM 开始工作在中心对齐模式时，PWM 时基周期寄存器（PTPER）的值被装入 PWM 时基寄存器（PTMR）且 PTMR 被自动配置为开始向下计数。这样可以确保不会同时产生所有的 PWM 信号。

图 13-15： 启动中心对齐的 PWM



13.6.5 互补的 PWM 操作

如图 13-16 所示， PWM 的互补工作模式对于驱动工作在半桥配置下的一个或多个功率开关很有用。这种逆变器结构常见于三相感应电机、无刷直流电机或三相不间断电源 （UPS）控制应用中。

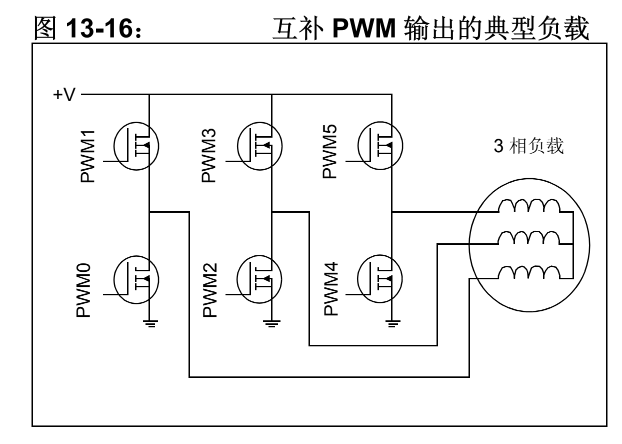
一对互补的 PWM 信号分别控制上 / 下功率开关对。当切换器件时，可选择插入死区时间（此时两路输出均短时无效）（见第 **13.7 节 " 死区时间发生器 "**）。

在互补模式中，占空比比较单元按如下方式被分配给 PWM 输出 ：

- PDC0 寄存器控制 PWM1/PWM0 输出
- PDC1 寄存器控制 PWM3/PWM2 输出
- PDC2 寄存器控制 PWM5/PWM4 输出

PWM1/3/5 是主 PWM，由 PDCx 寄存器控制， PWM0/2/4 则是与之互补的输出。当使用 PWM 控制半桥电路时，奇数编号的 PWM 用于控制上面的功率开关，而偶数编号的 PWM 则用于控制下面的开关。

图 13-16： 互补 PWM 输出的典型负载



通过将 PWMCON0 寄存器中相应的 PMODx 位清零来将相应的 PWM I/O 引脚对设定为互补模式。默认情况下， PWM I/O 引脚在所有复位条件下被设置为互补模式。

 超前信息

13.7 死区时间发生器

强烈建议在使用 PWM 的互补模式控制半桥电路的上部 / 下部开关的电源逆变器应用中插入死区时间。插入死区时间会使两路输出均短暂无效。这样可以避免在功率器件改变状态期间由于 TON 和 TOFF 特性而使两路输出发生混叠。

因为功率输出器件不可能瞬时完成切换，所以必须在禁止互补对中的一路 PWM 输出和导通另一个晶体管之间提供一定的时间。PWM 模块允许对死区时间进行编程。后面的小节将详细说明死区时间电路。

13.7.1 插入死区时间

PWM模块的每对互补输出都有一个6位的向下计数器，用于产生死区时间。如图 13-17 所示，每个死区时间单元都有与占空比比较输出相连的上升沿和下降沿检测器。当检测到 PWM 信号的边沿时，死区时间就会装入定时器。根据是上升沿还是下降沿，互补输出对中一路信号的转换将会延时直到定时器的计数值减小至零。图 13-18 显示了将死区时间插入一对 PWM 输出的时序图。



图 13-17： 一对 **PWM** 输出的死区时间控制单元框图

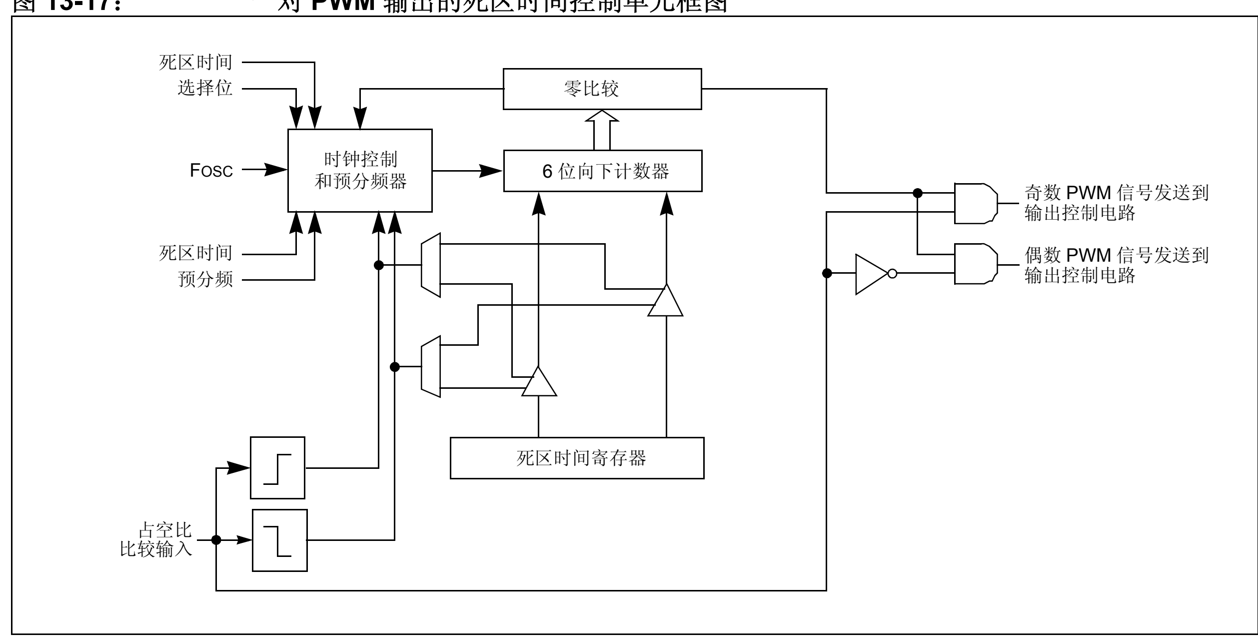
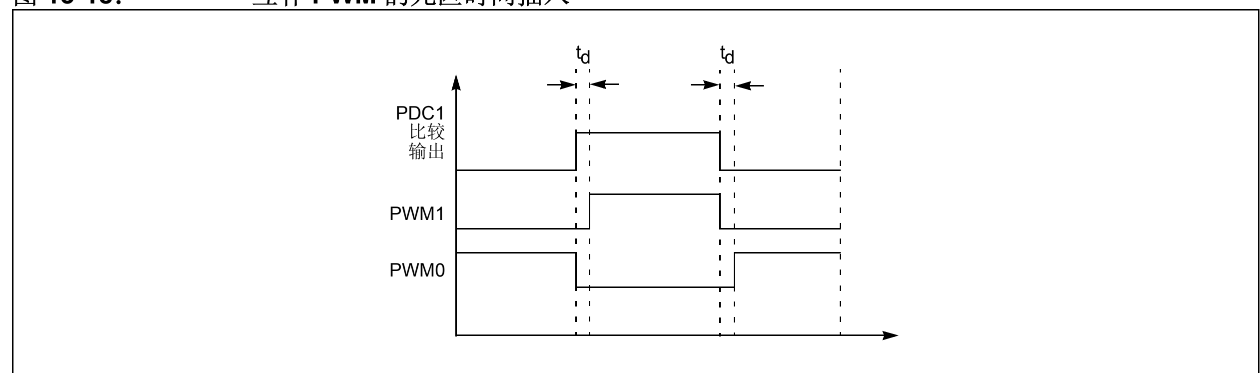


图 13-18： 互补 **PWM** 的死区时间插入



寄存器 13-5： DTCON：死区时间控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-6 **DTPS1:DTPS0**：死区时间单元 A 预分频比选择位
11 = 死区时间单元的时钟频率为 Fosc/16
10 = 死区时间单元的时钟频率为 Fosc/8
11 = 死区时间单元的时钟频率为 Fosc/4
11 = 死区时间单元的时钟频率为 Fosc/2

bit 5-0 **DT5:DT0**：死区时间单元的无符号 6 位死区时间值位

13.7.2 死区时间范围

通过在 DTCON 寄存器中指定输入时钟预分频值以及 6 位无符号值，可以选择死区时间量。提供了四种输入时钟预分频比选项，可用于根据器件的工作频率选择合适的死区时间范围。可用的时钟预分频比有 Fosc/2、Fosc/4、Fosc/8 和 Fosc/16，可以使用 DTCON 寄存器中的 DTPS1:DTPS0 控制位进行选择。

在选择了合适的预分频值后，可以通过将 6 位无符号值装入 DTCON<5:0> 调整死区时间。发生下列任一事件时，死区时间单元的预分频器将清零：

- 当发生占空比比较边沿事件时装载减计数定时器；
- 写入 DTCON 寄存器；或
- 任何器件复位。

13.7.3 死区时间计数器递减

死区时间计数器根据以下条件以任一 Q 时钟作为时钟源。

1. 在以下情况下，死区时间计数器在 Q1 递减：
   - 设置 DTPS 位使死区时间预分频比为以下任一值：Fosc/4、Fosc/8 或 Fosc/16
   - 设置 PWM 时基预分频比位（PTCKPS<1:0>）使预分频比为以下任一值：Fosc/16、Fosc/64 或 Fosc/256
2. 当 PWM 时基预分频比被设置为 1:1（PTCKPS<1:0> = 00，Fosc/4）且死区时间计数器的时钟频率为 Fosc/2（DTPS<1:0> = 00）时，死区时间计数器的时钟信号由一对 Q 时钟提供。
3. 根据占空比寄存器的两个 LSb，死区时间计数器可在每两个 Q 时钟递减一次：
   - 如果在 Q1 或 Q3 发生 PWM 占空比匹配，则死区时间计数器在每个 Q1 和 Q3 递减
   - 如果在 Q2 或 Q4 发生 PWM 占空比匹配，则死区时间计数器在每个 Q2 和 Q4 递减
4. 当将 DTPS<1:0> 设置为其他的死区时间预分频比（即 Fosc/4、Fosc/8 或 Fosc/16），并且 PWM 时基的预分频比被设置为 1:1 时，死区时间计数器在发生 PWM 占空比匹配的相应 Q 时钟上递减。

实际死区时间根据 DTCON 寄存器的值用下列公式计算：

死区时间 = 死区时间值 /(FOSC/ 预分频值)

表13-3给出了死区时间随选定输入时钟预分频值和器件工作频率变化的范围示例。

表 **13-3**：　死区时间范围示例

Fosc（MHz）	**MIPS**	预分频值选择	最小死区时间	最大死区时间
40	10	Fosc/2	50 ns	3.2 µs
40	10	Fosc/4	100 ns	6.4 µs
40	10	Fosc/8	200 ns	12.8 µs
40	10	Fosc/16	400 ns	25.6 µs
32	8	Fosc/2	62.5 ns	4 µs
32	8	Fosc/4	125 ns	8 µs
32	8	Fosc/8	250 ns	16 µs
32	8	Fosc/16	500 ns	32 µs
25	6.25	Fosc/2	80 ns	5.12 µs
25	6.25	Fosc/4	160 ns	10.2 µs
25	6.25	Fosc/8	320 ns	20.5 µs
25	6.25	Fosc/16	640 ns	41 µs
20	5	Fosc/2	100 ns	6.4 µs
20	5	Fosc/4	200 ns	12.8 µs
20	5	Fosc/8	400 ns	25.6 µs
20	5	Fosc/16	800 ns	51.2 µs
10	2.5	Fosc/2	200 ns	12.8 µs
10	2.5	Fosc/4	400 ns	25.6 µs
10	2.5	Fosc/8	800 ns	51.2 µs
10	2.5	Fosc/16	1.6 µs	102.4 µs
5	1.25	Fosc/2	400 ns	25.6 µs
5	1.25	Fosc/4	800 ns	51.2 µs
5	1.25	Fosc/8	1.6 µs	102.4 µs
5	1.25	Fosc/16	3.2 µs	204.8 µs
4	1	Fosc/2	0.5 µs	32 µs
4	1	Fosc/4	1 µs	64 µs
4	1	Fosc/8	2 µs	128 µs
4	1	Fosc/16	4 µs	256 µs

13.7.4　死区时间失真

> 注　**1：** 对于 PWM 占空比较小的情况，死区时间相对于有效 PWM 时间的比例可能会变大。在这种情况下，插入的死区时间可能会导致 PWM 模块产生的波形失真。通过保持 PWM 占空比至少是死区时间的三倍以上，用户可以确保死区时间失真最小。当占空比等于或接近 100% 的情况下，也会有同样的影响。在实际应用中使用的最大占空比应该选择使信号的无效时间至少比死区时间大三倍以上。如果死区时间大于或等于其中一对 PWM 输出的占空比，则该对 PWM 输出信号在整个周期内都会保持无效。
>
> **2：** 当使能 PWM 时修改 DTCON 的死区时间值可能会导致不良后果。在修改死区时间值前应该禁止 PWM （PTEN = 0）。

13.8　独立的 **PWM** 输出

独立的 PWM 模式用于驱动开关磁阻电机一个绕组的负载（见图 13-19）。当 PWMCON0 寄存器中相应的 PMODx 位置 1 时，特定的 PWM 输出对会被配置为独立输出模式。当模块工作在独立的 PWM 模式下时，不会在 PWM I/O 引脚信号之间插入死区时间，并且允许两个 I/O 引脚的输出同时有效。此模式还用于驱动步进电机。

13.8.1　独立 PWM 模式下的占空比分配

在独立的 PWM 模式下，给定输出对中的两个 PWM 输出引脚均连有一个占空比发生器。奇数和偶数 PWM 输出引脚都由同一个 PWM 占空比发生器驱动。PWM1 和 PWM0 由使用 PDC0 寄存器设置占空比的 PWM 通道驱动，相应的，PWM3 和 PWM2 对应的是 PDC1，PWM5 和 PWM4 对应的是 PDC2（见图 13-3 和寄存器 13-3）。

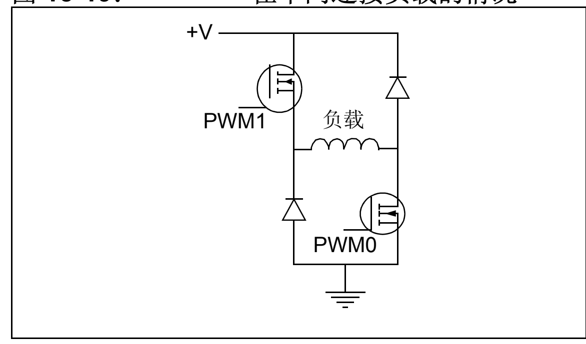
13.8.2 PWM 通道改写

使用 OVDCOND 和 OVDCONS 寄存器中相应的位可以手动改写每路 PWM 通道的 PWM 输出信号。对于每个工作在独立模式下的 PWM 输出引脚，用户可对输出进行如下配置：

- I/O 引脚输出 PWM 信号
- I/O 引脚无效
- I/O 引脚有效

如需有关改写功能的详细信息，请参见**第 13.10 节 “PWM 输出改写 ”**。

图 13-19： 在中间连接负载的情况



13.9 单脉冲 PWM 操作

仅在边沿对齐模式下，PWM 模块才可工作在单脉冲 PWM 模式下。在此模式下，PWM 模块将产生单脉冲输出。当 PTCON0 寄存器中 PTMOD1:PTMOD0 位设置为 01 时，模块被配置为单脉冲工作模式。此工作模式对于驱动某些类型的 ECM 很有用。

在单脉冲模式下，当 PTEN 位置 1 时，PWM I/O 引脚被驱动为有效状态。当 PWM 定时器与占空比寄存器匹配时，PWM I/O 引脚被驱动为无效状态。当 PWM 定时器与 PTPER 匹配时，PTMR 寄存器清零，所有的有效 PWM I/O 引脚均被驱动为无效状态，PTEN 位清零，并且如果相应的中断允许位置 1，还会产生一个中断。

注：	输出单个脉冲后，PTPER 和 PDCx 的值保持不变。要产生另一个单脉冲周期，只需要将 PTEN 位置 1。

13.10 PWM 输出改写

PWM 输出改写位可以让用户手动将 PWM I/O 引脚驱动为指定的逻辑状态，而不受占空比比较单元的影响。当控制各种 ECM （例如 BLDC 电机）时，PWM 改写位是很有用的。

OVDCOND 和 OVDCONS 寄存器用于定义 PWM 的改写选项。OVDCOND 寄存器包含 6 个位（POVD5:POVD0），这 6 个位决定要改写的是哪个 PWM I/O 引脚。OVDCONS 寄存器也包含 6 个位（POUT5:POUT0），它们用于决定被改写的 PWM I/O 引脚的状态。

POVD 位是低电平有效控制位。当 POVD 位置 1 时，相应的 POUT 位对 PWM 输出没有影响。换句话说，当 POVD 位置 1 时，与其对应的引脚的输出占空比由 PDCx 寄存器决定。当某个 POVD 位清零时，相应的 PWM I/O 引脚的输出将由 POUT 位的状态决定。当 POUT 位置 1 时，PWM 引脚将被驱动为其有效状态。当POUT位清零时，PWM引脚将被驱动为其无效状态。

13.10.1 互补的输出模式

当一对 PWM I/O 引脚工作在互补模式 （PMODx = 0）下时，对偶数 PWM I/O 引脚的改写有一定的限制。在互补模式下，如果通过将相应的 POVD 位清零以及设置 OVDCOND 和 OVDCONS 寄存器，将偶数编号的引脚驱动为有效，但实际的输出信号将被强制与该引脚对中奇数编号引脚的输出互补 （详细信息请见图 13-2）。

13.10.2 改写同步

如果 PWMCON1 寄存器中的 OSYNC 位置 1，所有通过 OVDCOND 和 OVDCONS 寄存器改写的输出都会与 PWM 时基同步。下列情况下会发生同步输出改写：

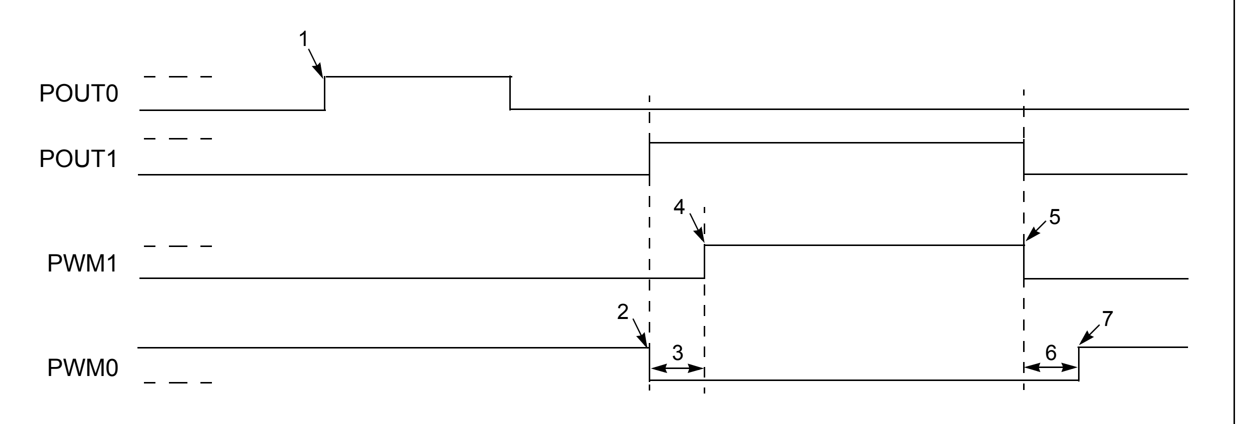
- 如果 PWM 处于边沿对齐模式下，当 PTMR 为 0 时发生同步。
- 如果 PWM 处于中心对齐模式下，当 PTMR 为 0 时，或 PTMR 的值与 PTPER 匹配时发生同步。

注	1：在互补模式下，当奇数通道的输出有效时，偶数通道不能由故障或改写事件强制设置为有效。偶数通道的输出始终与奇数通道的输出互补，当奇数通道被驱动为其有效状态前，将插入一段死区时间（见图 13-20）。 2：即使 PWM 通道处于改写模式，也会插入死区时间。

 超前信息



图 13-20： 互补模式下的改写位



假设：POVD0 = 0; POVD1 = 0; PMOD0 = 0

1. 在互补模式中偶数改写位不起作用。
2. 奇数改写位起效，导致偶数 PWM 输出无效。
3. 插入死区时间。
4. 奇数 PWM 在死区时间结束后有效。
5. 奇数改写位失效，导致奇数 PWM 失效。
6. 插入死区时间。
7. 偶数 PWM 在死区时间结束后有效。

13.10.3 输出改写示例

图 13-21 给出了使用 PWM 输出改写功能时，可能会产生的波形示例。此图显示了 BLDC 电机的 6 步换相序列。如图 13-16 所示，电机由三相逆变器驱动。当检测到相应的转子位置时， PWM 输出会切换到序列中的下一换相状态。在此例中， PWM 输出被驱动为特定的逻辑状态。表 13-4 列出了用来产生图 13-21 中信号的 OVDCOND 和 OVDCONS 寄存器值。

可将 PWM 占空比寄存器与 OVDCOND 和 OVDCONS 寄存器配合使用。占空比寄存器控制负载两端的平均电压， OVDCOND 和 OVDCONS 寄存器控制换相序列。图 13-22 显示了波形，而表 13-4 和表 13-5 则显示了用于产生相关信号的 OVDCOND 和 OVDCONS 寄存器的值。

寄存器 13-6： OVDCOND：输出改写控制寄存器

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-6 未用：读为 0

bit 5-0 **POVD5:POVD0**：PWM 输出改写位

1 = PWM I/O 引脚的输出由占空比寄存器的值和 PWM 时基控制
0 = PWM I/O 引脚的输出由对应的 POUTx 位控制

寄存器 13-7： OVDCONS：输出状态寄存器

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-6 未用：读为 0

bit 5-0 **POUT5:POUT0**：PWM 手动输出位

1 = 当对应的 PWM 输出改写位清零时 PWM I/O 引脚的输出有效
0 = 当对应的 PWM 输出改写位清零时 PWM I/O 引脚的输出无效

图 13-21： PWM 输出改写示例 1

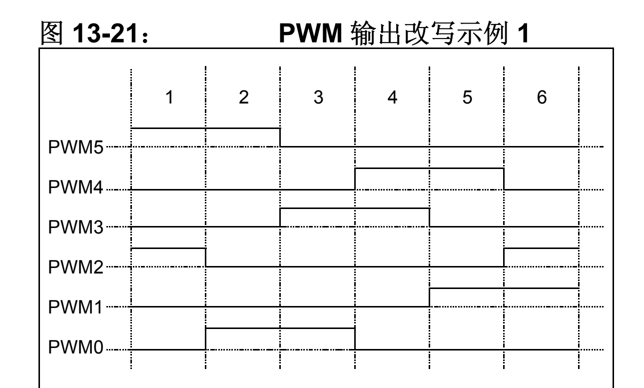


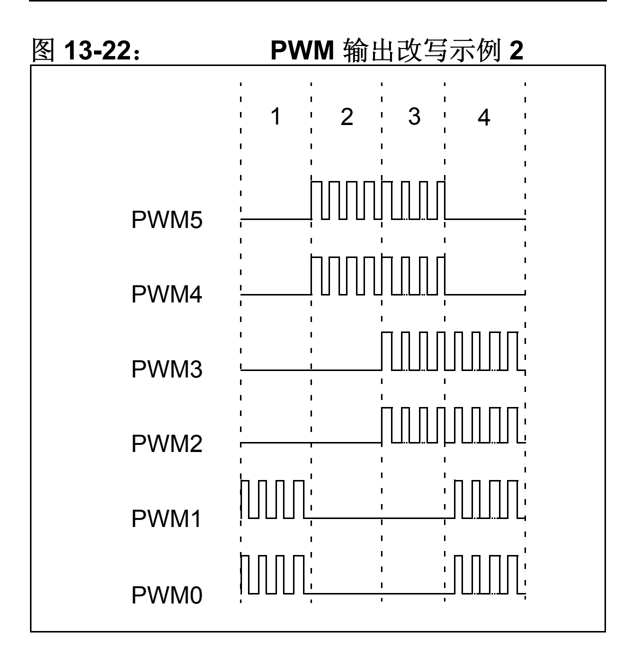
表 13-4： PWM 输出改写示例 1

状态	OVDCOND（POVD）	OVDCONS（POUT）
1	00000000b	00100100b
2	00000000b	00100001b
3	00000000b	00001001b
4	00000000b	00011000b
5	00000000b	00010010b
6	00000000b	00000110b

表 13-5： PWM 输出改写示例 2

状态	OVDCOND（POVD）	OVDCONS（POUT）
1	00000011b	00000000b
2	00110000b	00000000b
3	00111100b	00000000b
4	00001111b	00000000b

图 13-22： PWM 输出改写示例 2



13.11 PWM 输出和极性控制

在 CONFIG3L 寄存器中有 3 个与 PWM 模块相关的器件配置位，用于控制 PWM 输出引脚。它们是：

- HPOL
- LPOL
- PWMPIN

这 3 个配置位与 PWMCON0 寄存器中的 3 个 PWM 使能位（PWMEN2:PWMEN0）一起工作。配置位和PWM 使能位可以确保发生器件复位后 PWM 引脚处于正确的状态。

13.11.1 输出引脚控制

PWMEN2:PWMEN0 控制位根据应用的需要使能每个PWM 输出引脚。

所有的 PWM I/O 引脚都是通用 I/O。当一对引脚使能为PWM 输出时，控制引脚的 PORT 和 TRIS 寄存器被禁止。详情请参见图 13-23。

13.11.2 输出极性控制

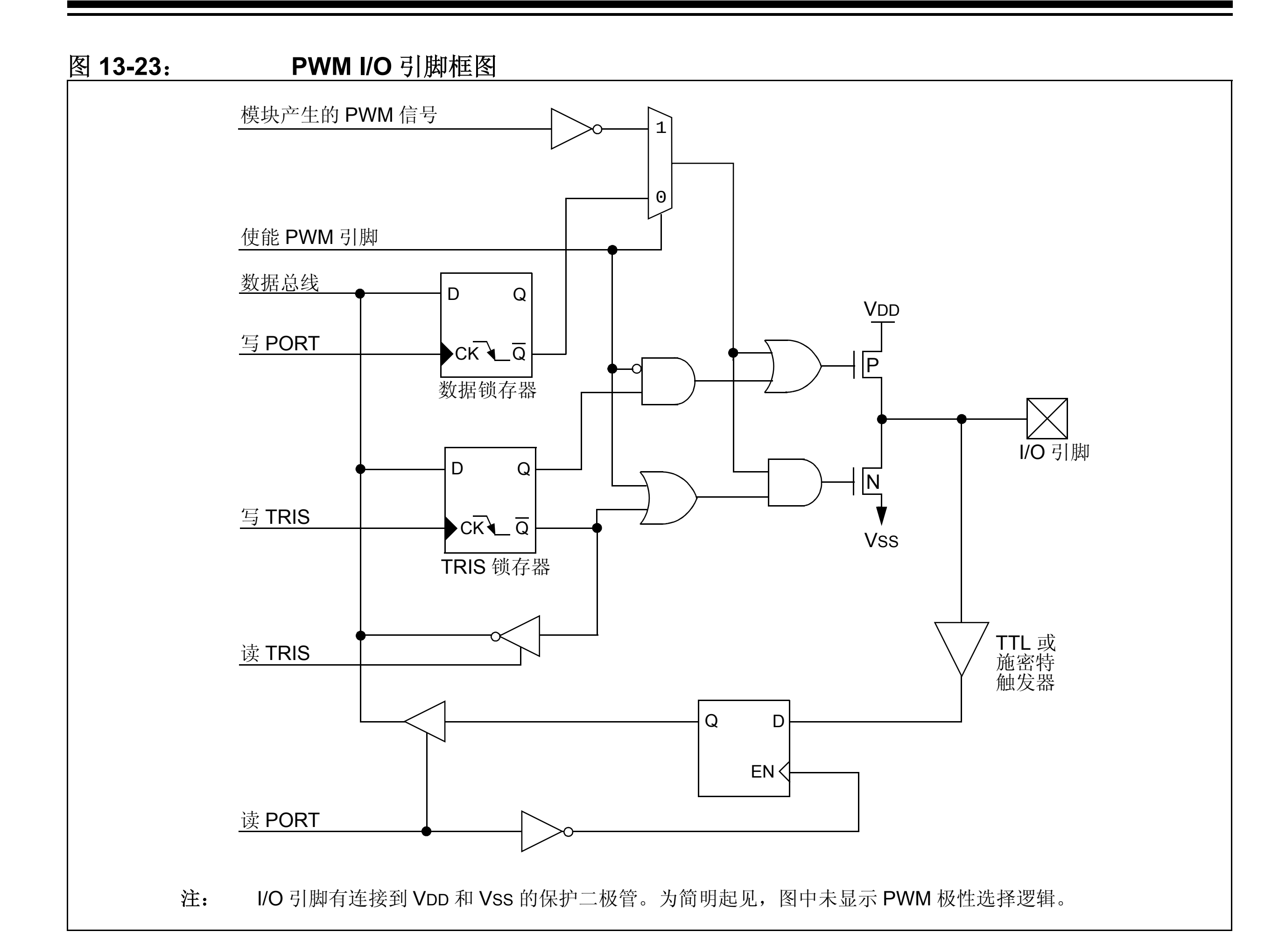
在编程器件时，通过 CONFIG3L 寄存器中的 HPOL 和LPOL 配置位设置 PWM I/O 引脚的极性。HPOL 配置位设置上半部分 PWM 输出（PWM1、PWM3 和 PWM5）的极性。当 HPOL 清零 （= 0）时，输出极性为高电平有效，而当 HPOL 置 1 （= 1）时，为低电平有效。

LPOL 配置位设置下半部分 PWM 输出（PWM0、PWM2 和 PWM4）的极性。与 HPOL 一样，当 LPOL 清零时这些引脚为高电平有效，当LPOL置1时为低电平有效。

PWM 模块产生的所有输出信号（包括由于故障输入或手动改写产生的信号）（见第 **13.10 节 “PWM 输出改写 ”**）均与极性控制位有关。

极性配置位的默认状态是使 PWM I/O 引脚的输出极性为高电平有效。

图 13-23： PWM I/O 引脚框图



注： I/O 引脚有连接到 VDD 和 VSS 的保护二极管。为简明起见，图中未显示 PWM 极性选择逻辑。

13.11.3 PWM 输出引脚复位状态

PWMPIN 配置位决定器件从复位状态恢复后，PWM 输出引脚是 PWM 输出引脚还是数字 I/O 引脚。如果 PWMPIN 配置位未经编程（默认），PWMEN2:PWMEN0 控制位会在器件复位时清零。此时，所有的 PWM 输出将为三态，并由相应的 PORT 和 TRIS 寄存器控制。如果 PWMPIN 配置位已编程为 `0`，PWMEN2:PWMEN0 控制位会在器件复位时被设置为 `100`。

所有的 PWM 引脚将使能为 PWM 输出引脚，并且其输出极性由 HPOL 和 LPOL 配置位定义。

13.12 PWM 故障输入

有一个与 PWM 模块相关的故障输入引脚。故障输入引脚的主要作用是禁止 PWM 输出信号并将它们驱动为无效状态。故障输入操作直接由硬件执行，因此当故障发生时，可以快速做出反应，迅速将 PWM 输出设置为无效状态以保护连接到 PWM 的功率器件。

PWM 故障输入引脚为 $\overline{\text{FLTA}}$，其信号可能有来自于 I/O 引脚、CPU 或其他模块。$\overline{\text{FLTA}}$ 引脚为低电平有效的输入引脚，因此可将多个故障源进行逻辑 “或” 运算后连接到同一个输入引脚。

FLTCONFIG 寄存器 （寄存器 13-8）用于设置 $\overline{\text{FLTA}}$ 输入。

注：	PWM 引脚的无效状态取决于 HPOL 和 LPOL 配置位的设置，这两个配置位定义 PWM 输出的有效和无效状态。

13.12.1 故障引脚使能位

通过将 FLTCONFIG 寄存器中的 FLTAEN 位置 1，可使能相应的故障输入。如果 FLTAEN 位清零，则故障输入信号对 PWM 模块没有影响。

 超前信息

13.12.2 故障输入模式

FLTCONFIG 寄存器中的 FLTAMOD 位决定 PWM I/O 引脚被故障输入改写后是否为失效状态。

FLTCONFIG 寄存器中的 FLTAS 位控制故障 A 输入的状态。

故障输入引脚有两种操作模式：

- **无效模式 （FLTAMOD = 0）**
  这是灾难性故障的管理模式。当在此模式中发生故障时， PWM 输出变为失效，并将保持这种状态直到故障被清除（故障输入引脚被驱动为高电平）并且相应的故障状态位由软件清零。故障状态位 （FLTAS）清零后， PWM 输出将在下一个 PWM 周期开始时立即使能。
- **逐周期模式 （FLTAMOD = 1）**
  当在此模式中发生故障时， PWM 输出变为失效。只要故障引脚保持为低电平， PWM 输出就将保持在定义的故障状态 （所有的 PWM 输出均无效）。故障引脚被驱动为高电平后， PWM 输出将在下一个 PWM 周期开始时返回正常工作状态，并且 FLTS 位会自动清零。

13.12.3 故障状态下的 PWM 输出

当处于故障状态 （即 $\overline{FLTA}$ 输入有效）时， PWM 输出信号被驱动为其无效状态。

13.12.4 调试模式下的 PWM 输出

当使用在线调试器（ICD）调试应用程序时，FLTCONFIG 寄存器中的 BRFEN 位用来控制遇到断点时是否模拟产生故障条件。将 BRFEN 设置为高电平允许在断点产生故障条件，这将使 PWM 输出驱动为无效状态。这样做是为了防止 PWM 引脚长时间地处于某一状态，从而避免与PWM输出相连的功率器件遭到破坏。

如果 BRFEN = 0，则禁止在断点处产生故障条件。

注：	如果在开发固件时使用了调试工具并且使用了大功率电路，则强烈建议使能断点故障条件模拟功能。在调试完成后准备对器件编程时， BRFEN 位可被禁止。

寄存器 13-8： FLTCONFIG：故障配置寄存器

R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
BRFEN	—	—	—	—	FLTAS	FLTAMOD	FLTAEN
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **BRFEN：**断点故障使能位
1 = 使能在断点产生故障条件
0 = 禁止产生故障条件

bit 6-3 未用：读为 0

bit 2 **FLTAS：**故障 A 状态位
1 = $\overline{FLTA}$ 为低电平：
如果 FLTAMOD = 0，该位由用户清零；
如果 FLTAMOD = 1，则当 $\overline{FLTA}$ 被拉高时，该位在新周期开始时自动清零。
0 = 无故障

bit 1 **FLTAMOD：**故障 A 模式位
1 = 逐周期模式：引脚在当前 PWM 周期余下的时间或 $\overline{FLTA}$ 拉高前保持无效； FLTAS 自动清零
0 = 无效模式：引脚在 FLTA 被拉高前保持无效 （灾难性故障）； FLTAS 只能由用户清零

bit 0 **FLTAEN：**故障 A 使能位
1 = 使能故障 A
0 = 禁止故障 A

13.13 PWM 更新锁定

对于复杂的 PWM 应用，用户在某个特定时间可能需要写多达 4 个占空比寄存器，以及时基周期寄存器PTPER。在某些应用中，一定要在新占空比和周期值被装载供模块使用前写入所有的缓冲寄存器。

也可以使能 PWM 更新锁定功能，这样用户可以指定新占空比缓冲器值有效的时间。通过将 PWMCON1 寄存器中的 UDIS 控制位置 1 可以使能 PWM 更新锁定功能。此位会影响所有占空比缓冲寄存器和 PWM 时基周期寄存器 PTPER。

要执行 PWM 更新锁定，需要：

1. 将 UDIS 位置 1。
2. 如果需要，写入所有的占空比寄存器和 PTPER。
3. 清零 UDIS 位以重新使能更新操作。
4. 这样， UDIS 位清零后，缓冲器值将装入真正的寄存器。这就完成了寄存器同步装载。

13.14 PWM 特殊事件触发器

PWM 模块具有特殊事件触发功能，可以使 A/D 转换与 PWM 时基同步。可以将 A/D 采样和转换时间编程为在 PWM 周期中的任何时刻发生。特殊事件触发功能使用户能够将获取 A/D 转换结果和更新占空比值之间的延时降至最短。

PWM 16 位特殊事件触发寄存器 SEVTCMP（高或低）和 PWMCON1 寄存器中的 5 个控制位用于控制其操作。把要发生特殊事件触发的 PTMR 值装入 SEVTCMP 寄存器对。PWMCON1 寄存器中的 SEVTDIR 位指定 PWM 时基处于连续向上 / 向下计数模式时的计数方向。

如果 SEVTDIR 位清零，特殊事件触发将在 PWM 时基的向上计数周期发生。如果 SEVTDIR 位置 1，特殊事件触发将在 PWM 时基的向下计数周期发生。SEVTDIR 位只影响 PWM 定时器处于连续向上 / 向下计数模式时的操作。

注：	只有 SEVTCMP 寄存器为非零值时才能产生特殊事件触发信号。

13.14.1 使能特殊事件触发器

PWM 模块总是会产生特殊事件触发脉冲。该脉冲信号也可以供 A/D 模块使用。详情请参见**第 15.0 节 "10 位模数转换器 （A/D）模块 "**。

13.14.2 特殊事件触发器后分频器

PWM 特殊事件触发器有一个后分频比为 1:1 至 1:16 的后分频器。通过写 PWMCON1 寄存器中的 SEVOPS3:SEVOPS0 控制位可以配置该后分频器。

对 SEVTCMP 寄存器对的任何写操作或器件复位都会使特殊事件触发器的输出后分频器清零。

 超前信息

表 13-6： 与电源控制 **PWM** 模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
IPR3	—	—	—	PTIP	—	—	—	—	43
PIE3	—	—	—	PTIE	—	—	—	—	43
PIR3	—	—	—	PTIF	—	—	—	—	43
PTCON0	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	43
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	43
PTMRL[(1)]	PWM 时基寄存器 （低 8 位）								43
PTMRH[(1)]	—	—	—	—	PWM 时基寄存器 （高 4 位）				43
PTPERL[(1)]	PWM 时基周期寄存器 （低 8 位）								43
PTPERH[(1)]	—	—	—	—	PWM 时基周期寄存器 （高 4 位）				43
SEVTCMPL[(1)]	PWM 特殊事件比较寄存器 （低 8 位）								44
SEVTCMPH[(1)]	—	—	—	—	PWM 特殊事件比较寄存器 （高 4 位）				44
PWMCON0	—	PWMEN2[(2)]	PWMEN1[(2)]	PWMEN0[(2)]	—	PMOD2	PMOD1	PMOD0	44
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	44
DTCON	DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0	44
FLTCONFIG	BRFEN	—	—	—	—	FLTAS	FLTAMOD	FLTAEN	43
OVDCOND	—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	44
OVDCONS	—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	44
PDC0L[(1)]	PWM 占空比寄存器 0L （低 8 位）								43
PDC0H[(1)]	—	—	PWM 占空比寄存器 0H （高 6 位）						43
PDC1L[(1)]	PWM 占空比寄存器 1L （低 8 位）								43
PDC1H[(1)]	—	—	PWM 占空比寄存器 1H （高 6 位）						43
PDC2L[(1)]	PWM 占空比寄存器 2L （低 8 位）								43
PDC2H[(1)]	—	—	PWM 占空比寄存器 2H （高 6 位）						43

图注： - = 未用，u = 不变。电源控制 PWM 不使用阴影单元。

注 1： 双重缓冲寄存器对。有关读写这些寄存器的说明请参见正文中的相关部分。

2： PWMEN 位的复位状态取决于 PWMPIN 配置位的设置。

注：

14.0 增强型通用同步 / 异步收发器（EUSART）

增强型通用同步 / 异步收发器（Enhanced Universal Synchronous Asynchronous Receiver Transmitter，EUSART）模块是两个串行 I/O 模块之一。（通常，也将USART称为串行通信接口或SCI。）可以将EUSART配置为能与 CRT 终端和个人计算机等外设通信的全双工异步系统；也可以将它配置成能够和A/D或D/A集成电路、串行 EEPROM 等外设通信的半双工同步系统。

增强型 USART 模块实现了更多的功能，包括自动波特率检测和校准，以及在接收到“同步间隔”字符和发送 12 位间隔字符时自动唤醒。这些功能使 EUSART 模块成为局域互连网络（Local Interconnect Network，LIN）总线系统理想的选择。

可将 EUSART 配置为以下几种工作模式：

- 带有以下功能的全双工异步模式：
  - 字符接收自动唤醒
  - 自动波特率校准
  - 12 位间隔字符发送
- 半双工同步主控模式（时钟极性可选）
- 半双工同步从动模式（时钟极性可选）

增强型 USART 的引脚与 PORTA 复用。为了将 RA2/TX/CK 和 RA3/RX/DT 引脚配置为 EUSART：

- SPEN（RCSTA<7>）位必须置 1（= 1）
- TRISA<3> 位必须置 1（= 1）
- TRISA<2> 位必须 置 1（= 1）

注：	EUSART 控制在需要时会自动将引脚从输入重新配置为输出。

增强型 USART 模块的操作由以下 3 个寄存器控制：

- 发送状态和控制寄存器（TXSTA）
- 接收状态和控制寄存器（RCSTA）
- 波特率控制寄存器（BAUDCON）

在后面几页的寄存器 14-1、寄存器 14-2和寄存器 14-3 中分别给出了上述寄存器的详细说明。

寄存器 14-1： TXSTA：发送状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **CSRC**：时钟源选择位
异步模式：
无关位。
同步模式：
1 = 主控模式 （时钟来自内部 BRG）
0 = 从动模式 （时钟来自外部时钟源）

bit 6 **TX9**：9 位发送使能位
1 = 选择 9 位发送
0 = 选择 8 位发送

bit 5 **TXEN**：发送使能位 (1)
1 = 使能发送
0 = 禁止发送

bit 4 **SYNC**：EUSART 模式选择位
1 = 同步模式
0 = 异步模式

bit 3 **SENDB**：发送间隔字符位
异步模式：
1 = 在下一次发送时发送 “同步间隔”字符 （在完成时用硬件清零）
0 = “同步间隔”字符发送完成
同步模式：
无关位。

bit 2 **BRGH**：高波特率选择位
异步模式：
1 = 高速
0 = 低速
同步模式：
在此模式下未使用。

bit 1 **TRMT**：发送移位寄存器状态位
1 = TSR 空
0 = TSR 满

bit 0 **TX9D**：发送数据的第 9 位
该位可以是地址 / 数据位或奇偶校验位。

注 1： 同步模式下 SREN/CREN 的优先级高于 TXEN。

 超前信息

寄存器 14-2：　RCSTA：接收状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7　**SPEN**：串行端口使能位
1 = 使能串行端口 （将 RX/DT 和 TX/CK 引脚配置为串行端口引脚）
0 = 禁止串行端口 （保持在复位状态）

bit 6　**RX9**：9 位接收使能位
1 = 选择 9 位接收
0 = 选择 8 位接收

bit 5　**SREN**：单字节接收使能位
异步模式：
无关位。
同步主控模式：
1 = 使能单字节接收
0 = 禁止单字节接收
此位在接收完成后清零。
同步从动模式：
无关位。

bit 4　**CREN**：连续接收使能位
异步模式：
1 = 使能接收器
0 = 禁止接收器
同步模式：
1 = 使能连续接收，直到使能位 CREN 清零 （CREN 比 SREN 优先级高）
0 = 禁止连续接收

bit 3　**ADDEN**：地址检测使能位
9 位异步模式 （RX9 = 1）：
1 = 当 RSR<8> 置 1 时，使能地址检测、允许中断并装载接收缓冲器
0 = 禁止地址检测、接收所有字节并且第 9 位可用作奇偶校验位
9 位异步模式 （RX9 = 0）：
无关位。

bit 2　**FERR**：帧错误位
1 = 帧错误 （可以通过读 RCREG 寄存器刷新并接收下一个有效字节）
0 = 无帧错误

bit 1　**OERR**：溢出错误位
1 = 溢出错误 （可以通过清零 CREN 位清除）
0 = 无溢出错误

bit 0　**RX9D**：接收数据的第 9 位
该位可以是地址 / 数据位或奇偶校验位，必须由用户固件计算得到。

寄存器 14-3： BAUDCON：波特率控制寄存器

R/W-0	R-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **ABDOVF**：自动波特率采集计满返回状态位
1 = 在自动波特率检测模式下发生了 BRG 计满返回 （必须由软件清零）
0 = 没有发生 BRG 计满返回

bit 6 **RCIDL**：接收操作空闲状态位
1 = 接收操作处于空闲状态
0 = 接收操作处于活动状态

bit 5 未用：读为 0

bit 4 **SCKP**：同步时钟极性选择位
异步模式：
在此模式下未使用。
同步模式：
1 = 空闲状态时钟 （CK）为高电平
0 = 空闲状态时钟 （CK）为低电平

bit 3 **BRG16**：16 位波特率寄存器使能位
1 = 16 位波特率发生器——SPBRGH 和 SPBRG
0 = 8 位波特率发生器——仅 SPBRG （兼容模式），忽略 SPBRGH 的值

bit 2 未用：读为 0

bit 1 **WUE**：唤醒使能位
异步模式：
1 = EUSART 将继续采样 RX 引脚——中断在下降沿产生，在下一个上升沿用硬件清零该位
0 = 未监测 RX 引脚或检测到了上升沿
同步模式：
在此模式下未使用。

bit 0 **ABDEN**：自动波特率检测使能位
异步模式：
1 = 在下一个字符使能波特率检测。需要收到 “同步” 字段 （55h），完成时由硬件清零
0 = 禁止波特率检测或检测已完成
同步模式：
在此模式下未使用。

14.1 波特率发生器 （BRG）

BRG 是一个专用的 8 位或 16 位发生器，支持 EUSART 的异步和同步模式。默认情况下，BRG 工作在 8 位模式下，将 BRG16 位（BAUDCON<3>）置 1 可选择 16 位模式。

SPBRGH:SPBRG 寄存器对控制自由运行定时器的周期。在异步模式下，BRGH（TXSTA<2>）位和 BRG16（BAUDCON<3>）位也控制波特率。在同步模式下，BRGH 位会被忽略。表 14-1 所示为不同 EUSART 模式的波特率计算公式，但仅适用于主控模式（由内部产生时钟信号）。

给定目标波特率和 Fosc 的情况下，可以使用表 14-1 中的公式计算 SPBRGH:SPBRG 寄存器中的最近似整数值，从而确定波特率误差。例 14-1 给出了计算示例。表 14-2给出了各种异步模式下典型的波特率和误差值。

使用高波特率 （BRGH = 1）或 16 位 BRG 有利于减小波特率误差，或者在快速振荡频率条件下实现低波特率。

向 SPBRGH:SPBRG 寄存器写入新值会引起 BRG 定时器复位 （或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。

14.1.1 在功耗管理模式下的操作

器件时钟用于产生所需的波特率。当进入一种功耗管理模式时，新时钟源可能会工作在与先前不同的频率下。这可能需要调整 SPBRG 寄存器对中的值。

14.1.2 采样

择多检测电路对 RX 引脚采样三次，以判定 RX 引脚上出现的是高电平还是低电平。

表 14-1： 波特率公式

配置位			BRG/EUSART 模式	波特率计算公式
SYNC	BRG16	BRGH		
0	0	0	8 位 / 异步	Fosc/[64 (n + 1)]
0	0	1	8 位 / 异步	Fosc/[16 (n + 1)]
0	1	0	16 位 / 异步	
0	1	1	16 位 / 异步	Fosc/[4 (n + 1)]
1	0	x	8 位 / 同步	
1	1	x	16 位 / 同步	

图注： x = 任意值， n = SPBRGH:SPBRG 寄存器对的值

例 14-1： 计算波特率误差

针对工作在异步模式下，工作频率 FOSC 为 16 MHz，采用 8 位 BRG，目标波特率为 9600 bps 的器件：

目标波特率 = FOSC/(64 ([SPBRGH:SPBRG] + 1))

求解 SPBRGH:SPBRG：

X = ((FOSC/ 目标波特率)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

计算得到的波特率 = 16000000/(64 (25 + 1))
= 9615

误差 = (波特率计算结果 – 目标波特率)/ 目标波特率
= (9615 – 9600)/9600 = 0.16%

表 14-2： 与波特率发生器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用 （读为 0）。 BRG 未使用阴影单元。

表 **14-3：** 异步模式的波特率

目标波特率（Kbps）	SYNC = 0，BRGH = 0，BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

目标波特率（Kbps）	SYNC = 0，BRGH = 0，BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

波特率（K）	SYNC = 0，BRGH = 1，BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

目标波特率（Kbps）	SYNC = 0，BRGH = 1，BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）	实际波特率（Kbps）	误差%	SPBRG 值（10 进制）
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

表 14-3： 异步模式的波特率 （续）

目标波特率（Kbps）	SYNC = 0，BRGH = 0， BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	0.300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1.201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2.403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

目标波特率（Kbps）	SYNC = 0，BRGH = 0， BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）
0.3	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

目标波特率（Kbps）	SYNC = 0， BRGH = 1， BRG16 = 1 或 SYNC = 1， BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

目标波特率（Kbps）	SYNC = 0， BRGH = 1， BRG16 = 1 或 SYNC = 1， BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）	实际波特率（Kbps）	误差 %	SPBRG 值（10 进制）
0.3	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.404	0.16	415	2.403	-0.16	207	2.403	-0.16	103
9.6	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

14.1.3 自动波特率检测

增强型 USART 模块支持波特率自动检测和校准。此功能仅在异步模式下当 WUE 位清零时有效。

只要接收到起始位并且 ABDEN 位已置 1，就会开始自动波特率检测过程（图 14-1）。波特率计算采用自平均的方式。

在自动波特率检测（Auto-Baud Rate Detect，ABD）模式下，BRG 的时钟是反向的。不是由 BRG 为 RX 输入信号提供时钟，而是由 RX 信号为 BRG 定时。在 ABD 模式下，内部波特率发生器被用作计数器来计算输入的串行字节流的位间隔时间。

一旦 ABDEN 位置 1，状态机就会将 BRG 清零并寻找起始位。为了正确计算位速率，自动波特率检测必须接收到一个值为 55h （ASCII 字符 U，也是 LIN 总线的同步字符）的字节。为了尽量减少输入信号不对称所造成的影响，在接收低位和高位的时间内都要进行测量。在起始位后，SPBRG 使用预先选择的时钟源在 RX 引脚信号的第一个上升沿开始计数。在 RX 引脚传输了 8 个位，或在检测到第 5 个上升沿后，会将在相应 BRG 周期内累加的值保存在 SPBRGH:SPBRG 寄存器对中。当第 5 个时钟周期出现时（应与停止位对应），ABDEN 位会自动清零。

如果发生了 BRG 计满返回 （从 FFFFh 到 0000h 的溢出），会在 ABDOVF 状态位（BAUDCON<7>）有所反映。当 BRG 计满返回时，该位由硬件置 1，用户也可用软件将其置 1 或清零。在发生计满返回事件后，继续保持 ABD 模式， ABDEN 位保持置 1 （图 14-2）。

在校准波特率周期时，BRG 寄存器时钟频率为预配置时钟频率的 1/8。请注意 BRG 时钟将由 BRG16 和 BRGH 位配置。无论 BRG16 的设置如何，SPBRG 和 SPBRGH 将被用作一个 16 位计数器。通过检查 SPBRGH 寄存器中的值是否为 00h，用户可以验证在 8 位模式下是否发生了进位。表 14-4 所示为 BRG 计数器的时钟速率。

当产生 ABD 序列时，EUSART 状态机保持在空闲状态。一旦在 RX 上检测到第 5 个上升沿，中断标志位 RCIF 就会置 1。需要读取 RCREG 中的值，来清除中断标志位 RCIF。应丢弃 RCREG 的值。

> 注 **1：** 如果 WUE 位与 ABDEN 位同时置 1，自动波特率检测会在间隔字符*之后*的字节开始。
>
> **2：** 用户需要判断进入的字符波特率是否处于所选 BRG 时钟源范围内。由于位错误率的原因，某些振荡频率和 EUSART 波特率的组合是无法实现的。在使用自动波特率检测功能时，必须综合考虑系统总的时序和通信波特率。

表 14-4： BRG 计数器时钟速率

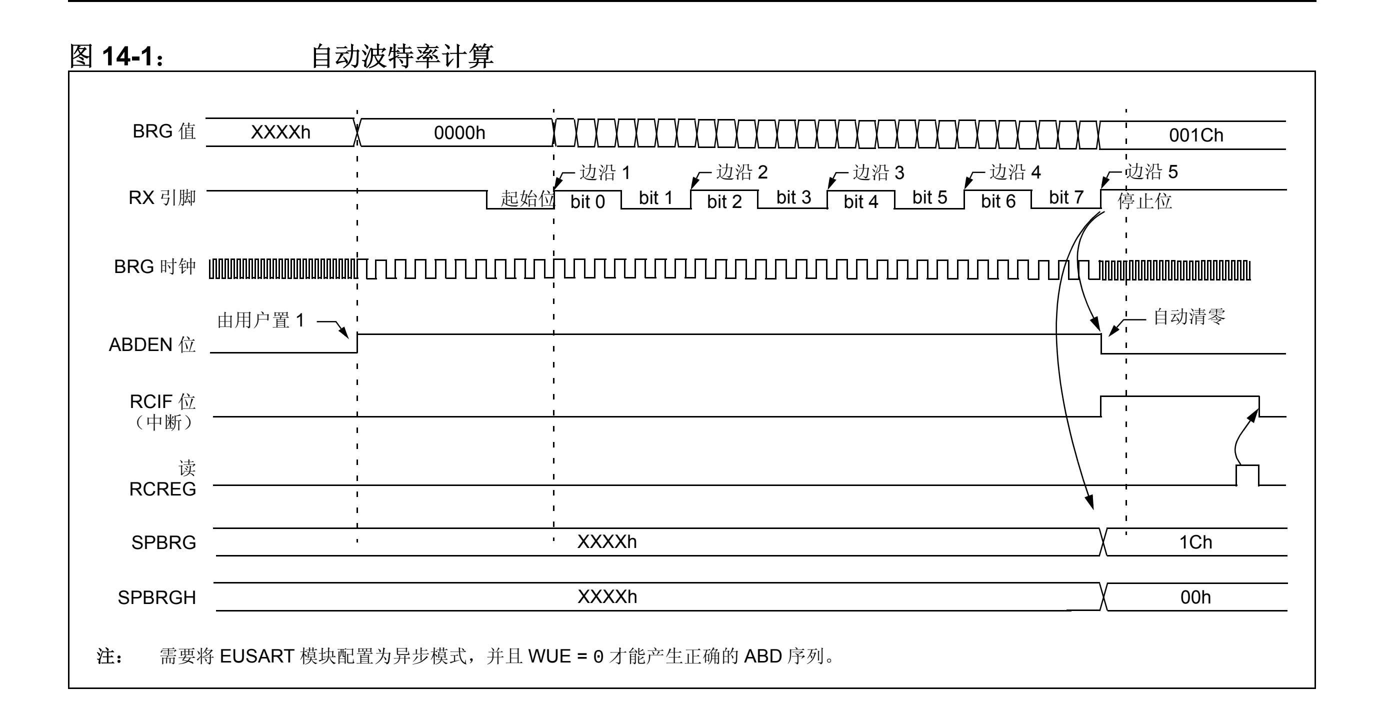
BRG16	BRGH	BRG 计数器时钟
0	0	FOSC/512
0	1	FOSC/128
1	0	FOSC/128
1	1	FOSC/32

注： 在产生 ABD 序列时，不管 BRG16 的设置如何，SPBRG 和 SPBRGH 被用作一个 16 位计数器。

14.1.3.1 ABD 和 EUSART 发送

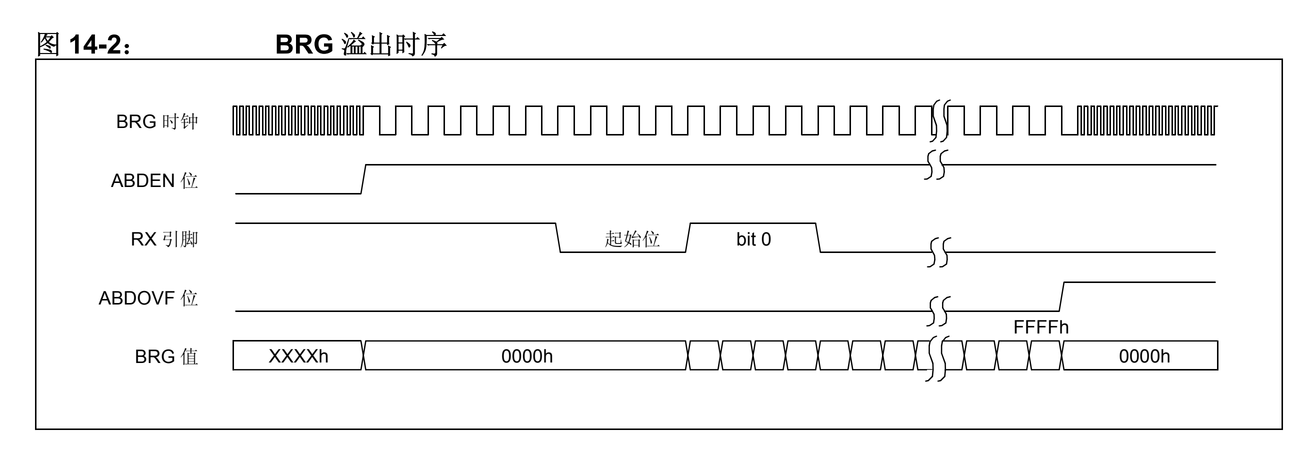
由于在 ABD 采集期间 BRG 时钟是反向的，因此在 ABD 期间不能使用 EUSART 发送器。这意味着只要 ABDEN 位置 1，就不能写入 TXREG。用户还应确保在发送期间 ABDEN 不能为置 1 状态，否则可能会导致无法预料的 EUSART 操作。

图 14-1： 自动波特率计算



注： 需要将 EUSART 模块配置为异步模式，并且 WUE = 0 才能产生正确的 ABD 序列。

图 14-2： BRG 溢出时序



14.2 EUSART 异步模式

通过将 SYNC 位（TXSTA<4>）清零可选择异步工作模式。在此模式下，EUSART 使用标准的不归零（Non-Return-to-Zero，NRZ）格式（1 个起始位、8 个或 9 个数据位和 1 个停止位）。最常用的格式为 8 个数据位。片上专用 8 位 /16 位波特率发生器可借助于振荡器产生标准波特率频率。

EUSART 首先发送和接收的是 LSb。EUSART 的发送器和接收器在功能上是独立的，但采用相同的数据格式和波特率。波特率发生器可以根据 BRGH 和 BRG16 位（TXSTA<2> 和 BAUDCON<3>）的设置值产生两种不同的波特率时钟，频率分别为移位速率的 16 倍或 64 倍。硬件不支持奇偶校验，但可以用软件实现，校验位保存在第 9 个数据位中。

当工作在异步模式下时，EUSART 模块包括以下重要组成部分：

- 波特率发生器
- 采样电路
- 异步发送器
- 异步接收器
- 同步间隔字符自动唤醒
- 12 位间隔字符发送
- 自动波特率检测

14.2.1 EUSART 异步发送器

图 14-3 显示了 EUSART 发送器的框图。发送器的核心是发送（串行）移位寄存器（Transmit Shift Register，TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG 中获取数据。 TXREG 寄存器中的数据由软件装入。在前一次装入的停止位被发送完成前，不会向 TSR 寄存器装入数据。一旦停止位发送完毕， TXREG 寄存器中的新数据（如果有的话）就会被装入 TSR。

一旦 TXREG 寄存器向 TSR 寄存器传输了数据（在 1 个 TCY 内发生），TXREG 寄存器就为空，同时标志位 TXIF（PIR1<4>）置 1。可以通过将中断允许位 TXIE（PIE1<4>）置 1 或清零来允许 / 禁止该中断。不管 TXIE 的状态如何，只要中断发生，TXIF 就会置 1 并且不能用软件清零。TXIF 不会在 TXREG 装入新数据时立即被清零，而是在装入指令后的第二个指令周期复位。因此在 TXREG 装入新数据后立即查询 TXIF，会返回无效值。

标志位 TXIF 指示的是 TXREG 寄存器的状态，而另一个位 TRMT（TXSTA<1>）则指示 TSR 寄存器的状态。TRMT 是只读位，它在 TSR 寄存器为空时置 1。TRMT 位与任何中断均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。

注	
1：	TSR 寄存器并未映射到数据存储器中，因此用户不能访问它。
2：	当使能位 TXEN 置 1 时，标志位 TXIF 也置 1。

设置异步发送的操作步骤如下：

1. 对 SPBRGH:SPBRG 寄存器进行初始化，以设置适合的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得目标波特率。
2. 通过将 SYNC 位清零、SPEN 位置 1，使能异步串行端口。
3. 若需要中断，将中断允许位 TXIE 置 1。
4. 若需要发送 9 位数据，将发送位 TX9 置 1。发送的第 9 位可以是地址位也可以是数据位。
5. 通过将 TXEN 位置 1 使能发送，此操作同时也会将 TXIF 位置 1。
6. 如果选择发送 9 位数据，应该将第 9 位数据装入 TX9D 位。
7. 将数据装入 TXREG 寄存器（开始发送）。
8. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 14-3： EUSART 发送框图

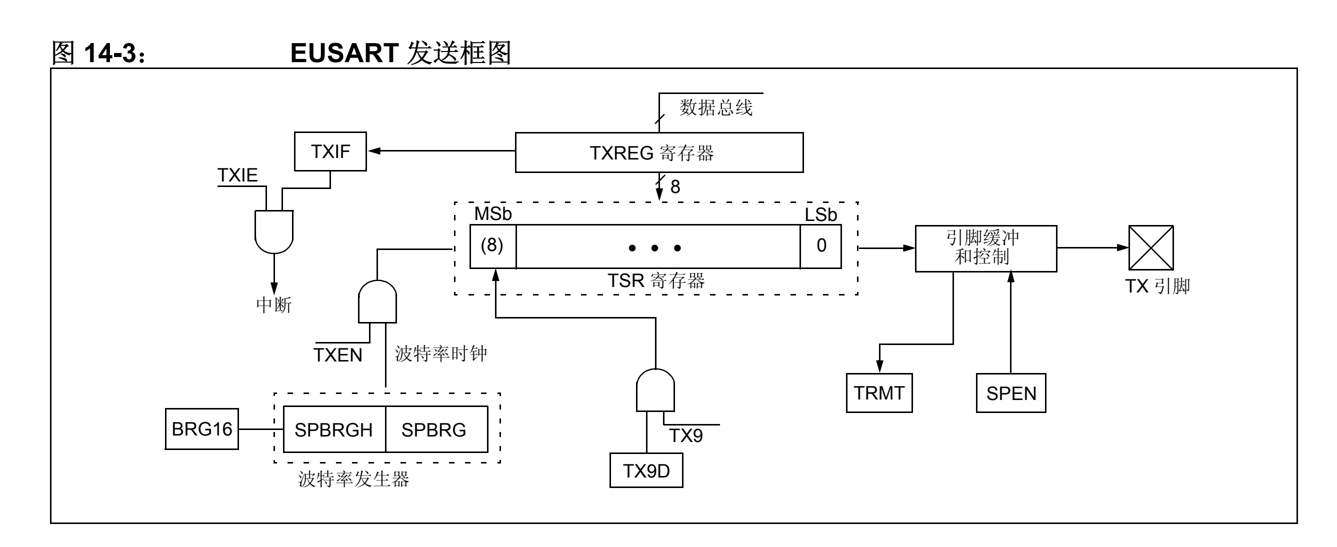


图 14-4： 异步发送

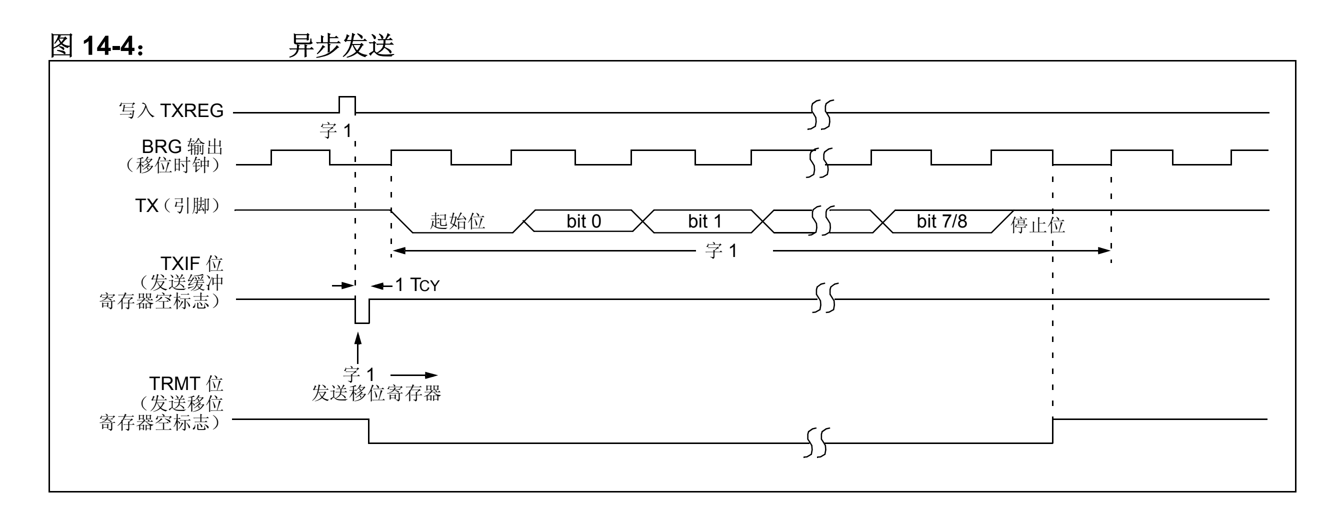
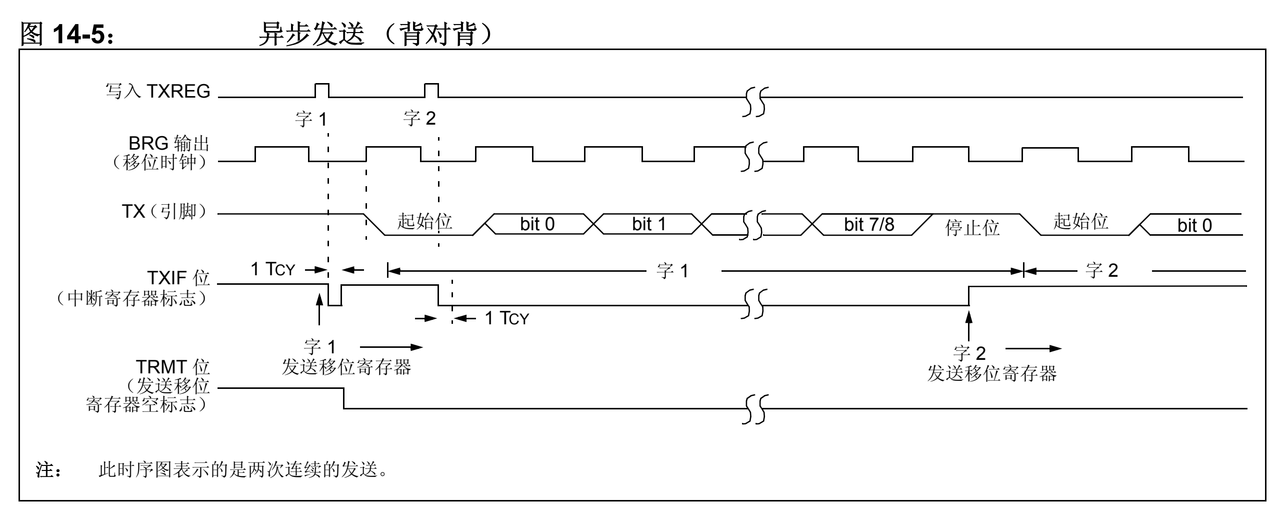


图 14-5： 异步发送 （背对背）



注： 此时序图表示的是两次连续的发送。

表 14-5： 与异步发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
TXREG	EUSART 发送寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用单元 （读为 0）。异步发送不使用阴影单元。

 超前信息

14.2.2 EUSART 异步接收器

图 14-6 显示了接收器的框图。在 RX 引脚上接收数据，并驱动数据恢复电路。数据恢复电路实际上是一个工作频率为 16 倍波特率的高速移位器，而主接收串行移位器的工作频率等于比特率或 Fosc。此模式通常用于 RS-232 系统。

设置异步接收操作的步骤如下：

1. 对 SPBRGH:SPBRG 寄存器进行初始化，以设置适合的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得目标波特率。
2. 通过将 SYNC 位清零、SPEN 位置 1，使能异步串行端口。
3. 若需要中断，将中断允许位 RCIE 置 1。
4. 若需要接收 9 位数据，将 RX9 位置 1。
5. 通过将 CREN 位置 1，使能接收。
6. 当接收完成时标志位 RCIF 将置 1，此时如果中断允许位 RCIE 已置 1，还将产生一个中断。
7. 读 RCSTA 寄存器以获取第 9 位数据（如果已使能），并判断接收过程中是否发生了错误。
8. 通过读RCREG寄存器来读取接收到的8位数据。
9. 如果发生错误，通过将 CREN 清零来清除错误。
10. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

14.2.3 设置带有地址检测功能的 9 位模式

此模式通常用于 RS-485 系统。按如下步骤设置带有地址检测功能的异步接收操作：

1. 对 SPBRGH:SPBRG 寄存器进行初始化，以设置适合的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得目标波特率。
2. 通过将 SYNC 位清零、SPEN 位置 1，使能异步串行端口。
3. 若需要中断，将 RCEN 位置 1 并使用 RCIP 位设置优先级。
4. 将 RX9 位置 1，使能 9 位接收操作。
5. 将 ADDEN 位置 1，使能地址检测。
6. 将 CREN 位置 1，使能接收。
7. 当接收完成时 RCIF 位将置 1。此时如果 RCIE 和 GIE 位已置 1，还将响应中断。
8. 读 RCSTA 寄存器判断在接收时是否发生了错误，同时读取第 9 位数据（如果适用）。
9. 读 RCREG 来判断是否正在对器件进行寻址。
10. 如果发生错误，将 CREN 位清零。
11. 如果已经找到了器件，将 ADDEN 位清零，允许接收到的所有数据进入接收缓冲器，并中断 CPU。

图 14-6： EUSART 接收框图

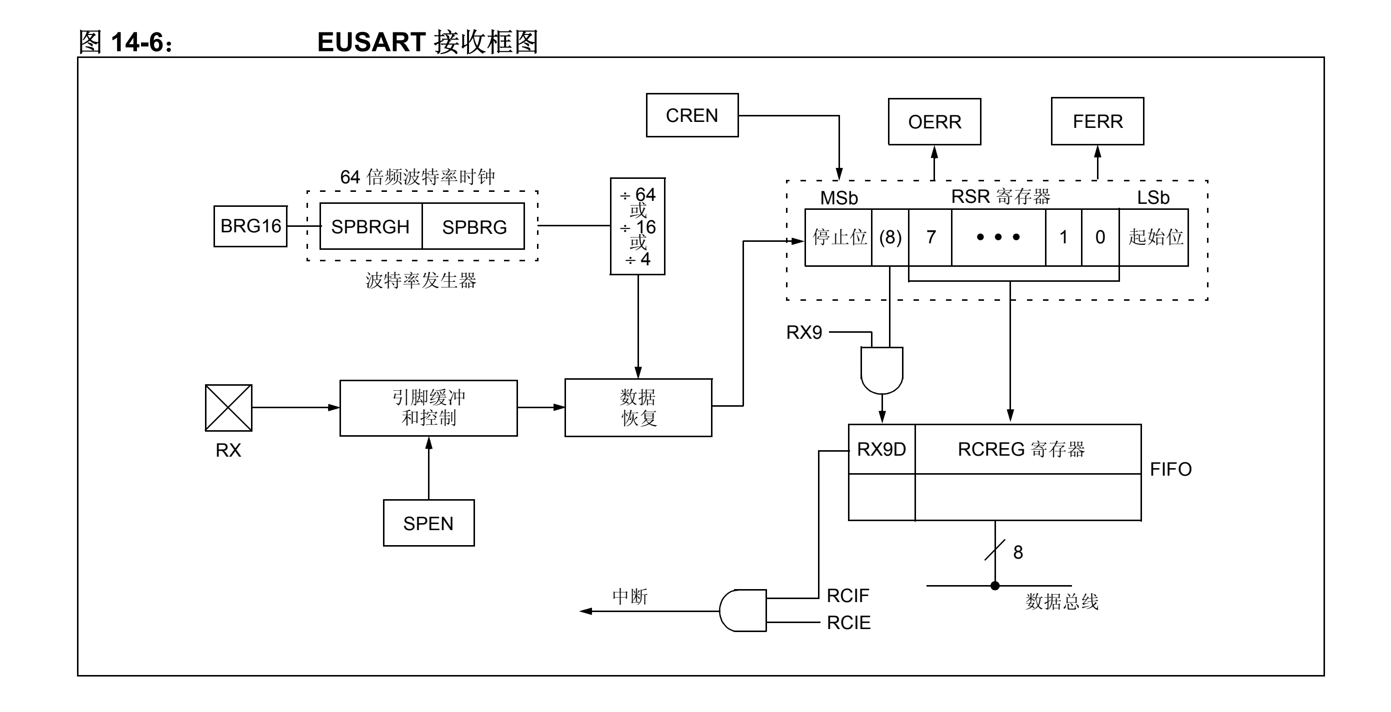
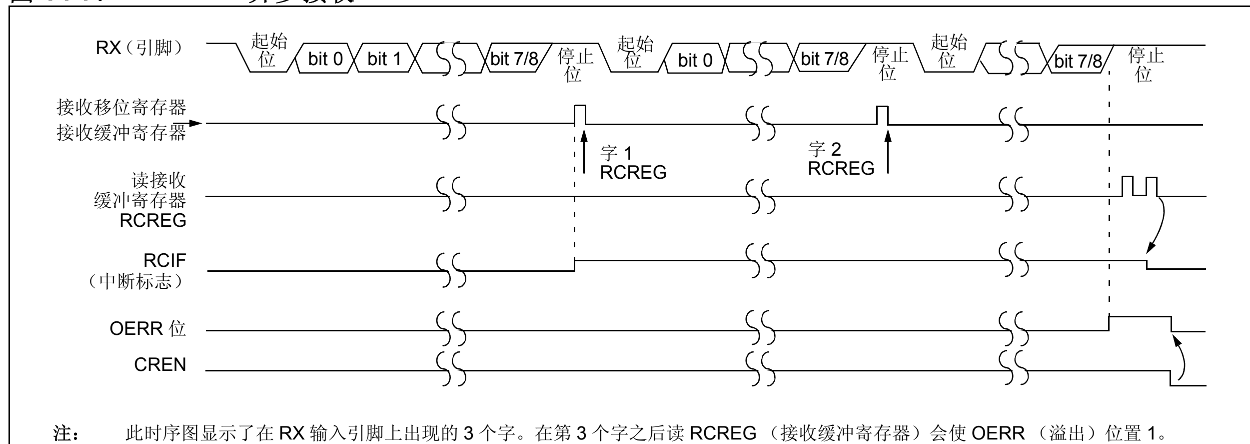


图 14-7：　　　　异步接收



注：　此时序图显示了在 RX 输入引脚上出现的 3 个字。在第 3 个字之后读 RCREG （接收缓冲寄存器）会使 OERR （溢出）位置 1。

表 14-6：　　　　与异步接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
RCREG	EUSART 接收寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注：　— = 未用单元 （读为 0）。异步接收不使用阴影单元。

14.2.4　同步间隔字符自动唤醒

在休眠模式下，EUSART 的所有时钟都会暂停。因此，波特率发生器处于非活动状态，并且无法进行正确的字节接收。自动唤醒功能允许在RX/DT线上有活动时唤醒控制器，该功能需要 EUSART 工作在异步模式下。

通过将 WUE 位 （BAUDCON<1>）置 1，使能自动唤醒功能。置 1 后，将禁止 RX/DT 上的典型接收操作，且 EUSART 保持在空闲状态并监视唤醒事件（与 CPU 运行模式无关）。唤醒事件是指RX/DT线上发生高电平到低电平的转换。（这与 “同步间隔”字符或 LIN 协议唤醒信号字符的启动条件一致。）

唤醒事件后，模块产生一个 RCIF 中断。在正常工作模式下，中断会与 Q 时钟同时产生 （图 14-8）；如果器件处于休眠模式，则两者不同步 （图 14-9）。通过读 RCREG 寄存器可清除中断条件。

唤醒事件后，在 RX 线上检测到由低电平向高电平的转换时，WUE 位自动清零。此时， EUSART 模块将从空闲状态返回正常工作模式，由此用户可知 “同步间隔”事件结束。

14.2.4.1 使用自动唤醒功能的注意事项

因为自动唤醒功能是通过检测RX/DT上的上升沿跳变实现的，所以在停止位前该引脚上任何的状态改变都可能会产生错误的结束信号并导致数据或帧错误。因此，为了确保正确的传输，必须首先发送全 0 字符。对于标准的 RS-232 器件，这可以是 00h （8 位），而对于 LIN 总线则是 000h （12 位）。

另外还必须考虑振荡器起振时间，尤其在采用起振延时较长的振荡器 （即， XT 或 HS 模式）的应用中更要注意这一点。“同步间隔”（或唤醒信号）字符必须足够长，并且跟有足够长的时间间隔，以便使选定振荡器有充足的时间起振并保证 EUSART 被正确初始化。

14.2.4.2 使用 WUE 位的注意事项

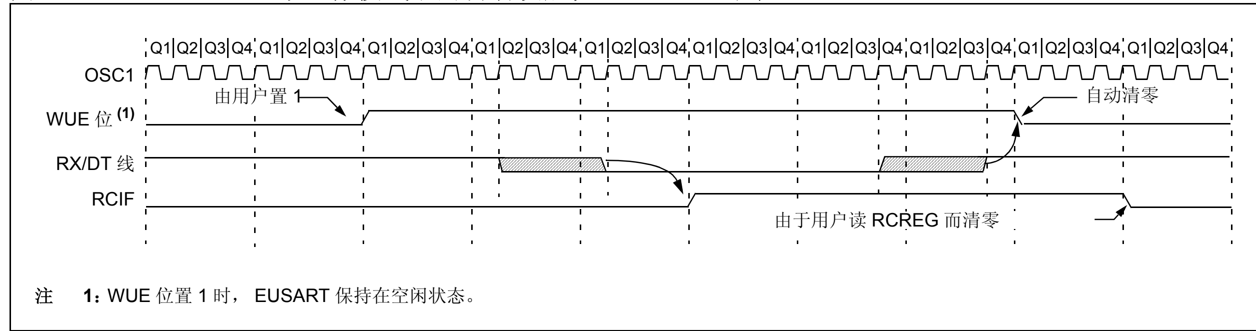
用 WUE 和 RCIF 事件的时序来判断接收数据的有效性可能会引起混淆。如前所述，将 WUE 位置 1 会使 EUSART 进入空闲模式。唤醒事件会产生一个接收中断并将 RCIF 位置 1。此后，当 RX/DT 出现上升沿时 WUE 位清零。然后通过读 RCREG 寄存器清除中断条件。一般情况下， RCREG 中的数据是无效数据，应该丢弃。

WUE 位清零 （或仍然置 1）且 RCIF 标志位置 1 并不能表明 RCREG 中接收的数据是完整的。用户还应该考虑使用固件验证是否完整地接收了数据。

要确保没有丢失有效数据，应检查RCIDL位来验证是否还在接收数据。如果不在接收数据，则可将 WUE 位置 1 随即器件进入休眠模式。

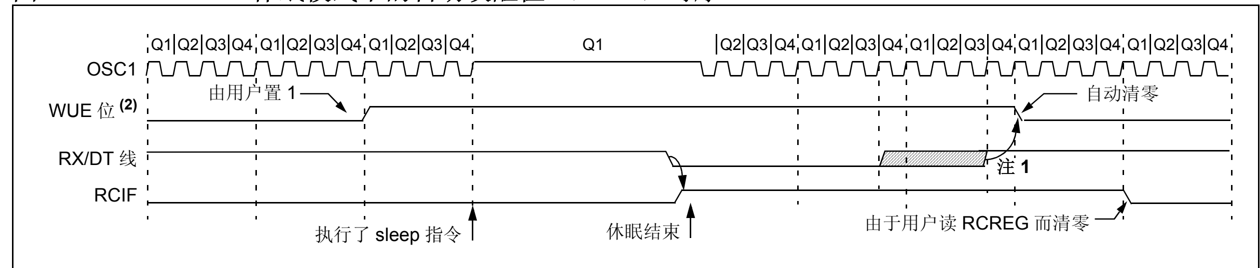


图 14-8： 正常工作模式下的自动唤醒位 （**WUE**）时序



注 1：WUE 位置 1 时， EUSART 保持在空闲状态。

图 14-9： 休眠模式下的自动唤醒位 （**WUE**）时序



注 1： 如果唤醒事件需要较长的振荡器预热时间，在振荡器准备就绪之前 WUE 位可能自动清零。此过程与 Q 时钟出现与否无关。
2： WUE 位置 1 时， EUSART 保持在空闲状态。

14.2.5 间隔字符序列

EUSART模块能够发送符合LIN总线标准的特殊间隔字符序列。发送的间隔字符包括 1 个起始位，后面跟有 12 个 0 位和一个停止位。当发送移位寄存器装有数据时，只要 SENDB 和 TXEN 位（TXSTA<3> 和 TXSTA<5>）置 1，就会发送帧间隔字符。请注意写入 TXREG 的数据值会被忽略，并会发送全 0。

在发送了相应的停止位后，硬件会自动将 SENDB 位清零。这样用户可以在发送完间隔字符（在 LIN 规范中通常是同步字符）后将下一个要发送的字节预先装入发送 FIFO。

请注意发送间隔字符时写入 TXREG 的数据值会被忽略。写入仅仅是为了启动正确的序列。

如其在正常发送操作中一样，TRMT 位表明发送正在进行还是处于空闲状态。关于间隔字符序列，请参见图 14-10。

14.2.5.1 间隔和同步发送序列

以下序列会发送一个报文帧头，包括一个间隔字符和其后的自动波特率同步字节。此发送序列适用于典型的 LIN 总线主控器件。

1. 将 EUSART 配置为所需的模式。
2. 将 TXEN 和 SENDB 位置 1 以设置间隔字符。
3. 将无效字符装入 TXREG，启动发送（该值会被忽略）。
4. 将 55h 写入 TXREG，以便把同步字符装入发送 FIFO 缓冲器。
5. 间隔字符发送后，硬件会将 SENDB 位复位。此时，同步字符会以预先配置的模式发送。

当 TXIF 指出 TXREG 为空以后，下一个数据字节会被写入 TXREG。

14.2.6 接收间隔字符

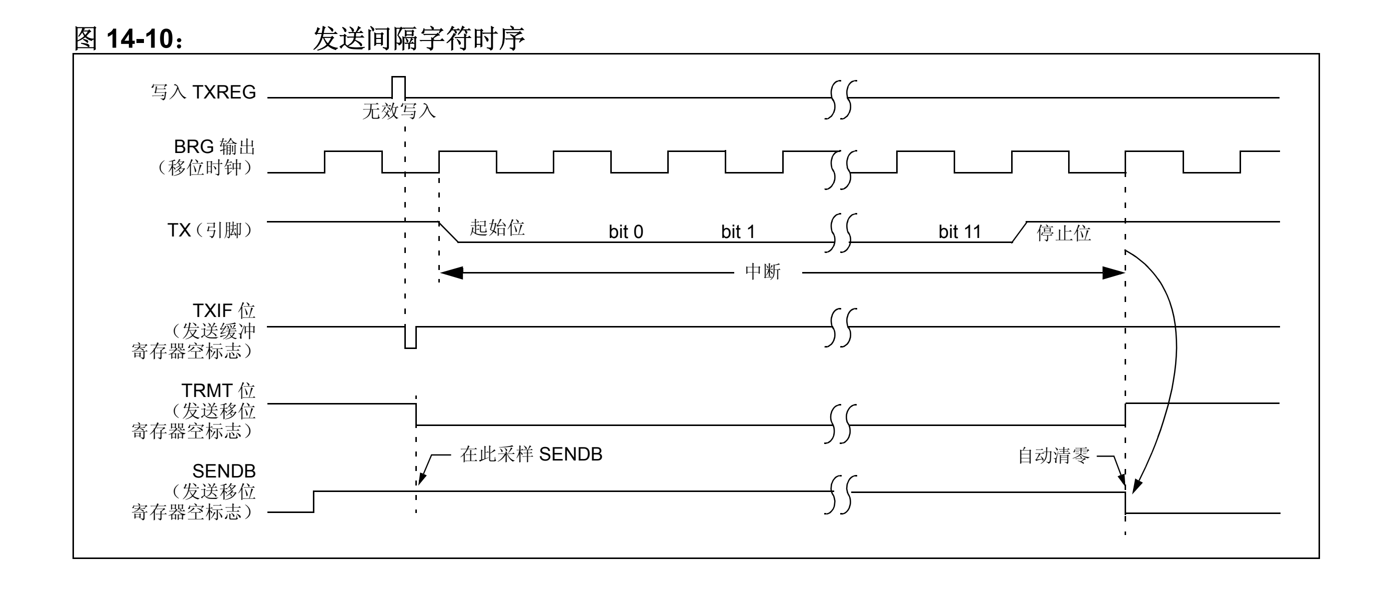
增强型 USART 模块接收间隔字符有两种方法。

第一种方法是强制将波特率配置为典型速度的 9/13。这可以使停止位在正确的采样点（对于间隔字符为起始位之后的 13 位，对于典型数据则是 8 个数据位）产生。

第二种方法使用**第 14.2.4 节“同步间隔字符自动唤醒”**中描述的自动唤醒功能。通过使能此功能，EUSART 将采样 RX/DT 引脚上电平的下两次跳变，产生一个 RCIF 中断，接收下一个数据字节，并在随后产生另一个中断。

请注意在间隔字符后，用户通常希望使能自动波特率检测功能。无论使用哪种方法，用户都可以在检测到 TXIF 中断时马上将 ABDEN 位置 1。

图 14-10： 发送间隔字符时序



 超前信息

14.3 EUSART 同步主控模式

将 CSRC 位（TXSTA<7>）置 1 可以进入同步主控模式。在此模式中，数据以半双工方式（即发送和接收不同时进行）发送。发送数据时，禁止接收，反之亦然。将 SYNC 位（TXSTA<4>）置 1 可进入同步模式。此外，应将使能位 SPEN（RCSTA<7>）置 1，分别把 TX 和 RX 引脚配置为 CK（时钟）和 DT（数据）线。

主控模式意味着处理器在 CK 时钟线上发送主控时钟信号。时钟极性是通过 SCKP 位（BAUDCON<4>）选择的。将 SCKP 置 1 是将空闲状态时的 CK 设为高电平，将该位清零则将空闲状态时的 CK 设为低电平。此选项支持将本模块与 Microwire 器件配合使用。

14.3.1 EUSART 同步主控发送

图 14-3 显示了 EUSART 发送器的框图。发送器的核心是发送（串行）移位寄存器（TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG 中获取数据。TXREG 寄存器中的数据由软件装入。在前一次装入数据的最后一位发送完成后，才向 TSR 寄存器装入新数据。一旦最后一位发送完成，就会将 TXREG 寄存器中的新数据（如果有的话）装入 TSR。

一旦 TXREG 寄存器向 TSR 寄存器传输了数据（在 1 个 TCY 内发生），TXREG 寄存器就为空，同时标志位 TXIF（PIR1<4>）置 1。可以通过将中断允许位 TXIE（PIE1<4>）置 1 或清零来允许或禁止该中断。TXIF 的设置与允许位 TXIE 的状态无关，且不能用软件清零。只有在新数据写入 TXREG 寄存器时，TXIF 才会复位。

TXIF 表示的是 TXREG 寄存器的状态，而另一个标志位 TRMT（TXSTA<1>）则表示 TSR 寄存器的状态。TRMT 是只读位，它在 TSR 寄存器为空时置 1。TRMT 位与任何中断均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。TSR 并未映射到数据存储器中，因此用户不能直接访问它。

设置同步主控发送操作的步骤如下：

1. 对 SPBRGH:SPBRG 寄存器进行初始化，以设置适合的波特率。按需要将 BRG16 位置 1 或清零，以获得目标波特率。
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行端口。
3. 若需要中断，将中断允许位 TXIE 置 1。
4. 若需要发送 9 位数据，将 TX9 位置 1。
5. 将 TXEN 位置 1，使能发送。
6. 如果选择发送 9 位数据，应该将第 9 位数据装入 TX9D 位。
7. 将数据装入 TXREG 寄存器，启动发送。
8. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 14-11： 同步发送时序

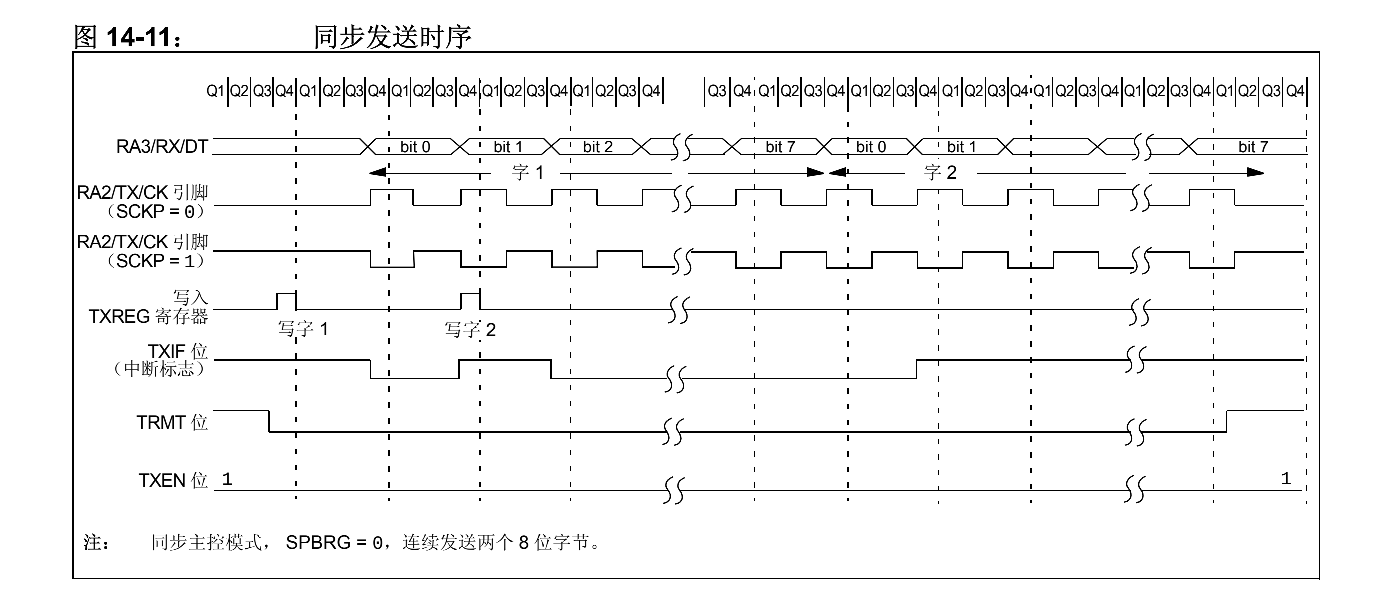


图 14-12： 同步发送时序 （由 TXEN 位控制）

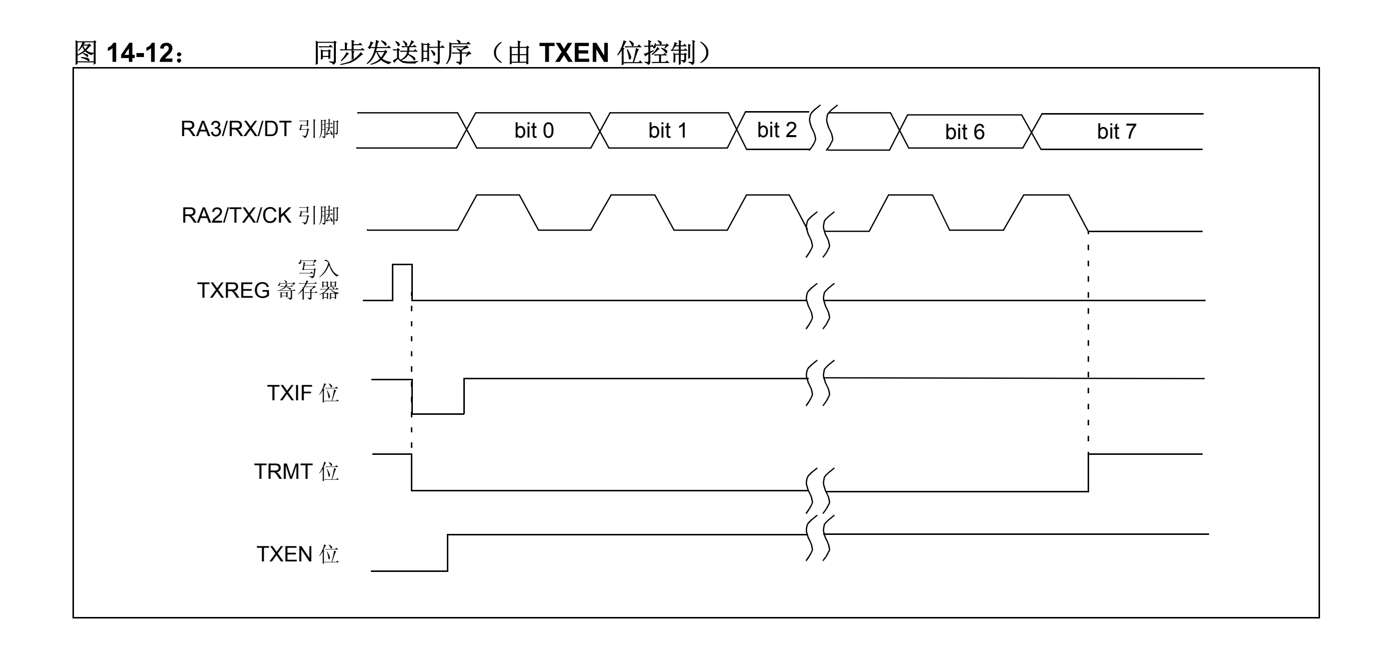


表 14-7： 与同步主控发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
TXREG	EUSART 发送寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用 （读为 0）。同步主控发送不使用阴影单元。

 超前信息

14.3.2 EUSART 同步主控接收

一旦选择了同步模式，只要将单字节接收使能位 SREN （RCSTA<5>）或连续接收使能位 CREN （RCSTA<4>）置 1，即可使能接收。在时钟的下降沿采样 RX 引脚上的数据。

如果将使能位 SREN 置 1，则只接收单个字。如果将使能位 CREN 置 1，则会连续接收数据，直到将 CREN 位清零。如果两个位均被置 1，则 CREN 具有优先权。

设置同步主控接收操作的步骤如下：

1. 对 SPBRGH:SPBRG 寄存器进行初始化，以设置适合的波特率。按需要将 BRG16 位置 1 或清零，以获得目标波特率。
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行端口。
3. 确保将 CREN 和 SREN 位清零。
4. 若需要中断，将中断允许位 RCIE 置 1。
5. 若需要接收 9 位数据，将 RX9 位置 1。
6. 若需要单字节接收，将 SREN 位置 1。若需要连续接收，将 CREN 位置 1。
7. 当接收完成时中断标志位 RCIF 将置 1，此时如果中断允许位 RCIE 已置 1，则还将产生一个中断。
8. 读 RCSTA 寄存器获取第 9 位数据（如果已使能），并判断在接收过程中是否发生了错误。
9. 通过读RCREG寄存器来读取接收到的8位数据。
10. 如果发生错误，将 CREN 位清零以清除错误。
11. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

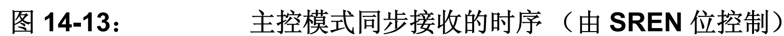
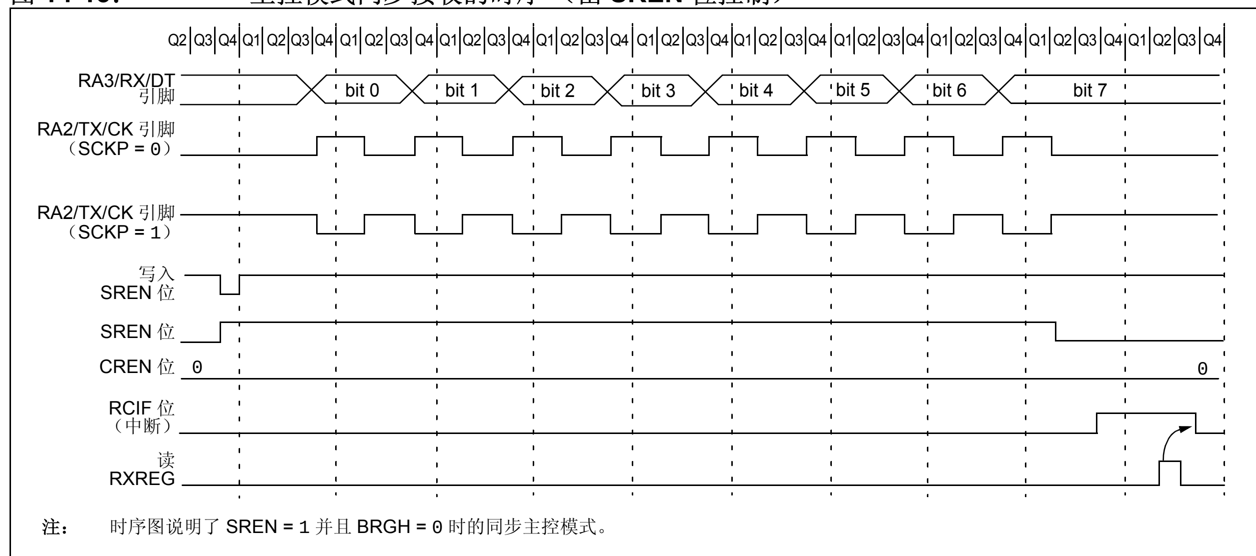


图 14-13： 主控模式同步接收的时序 （由 **SREN** 位控制）



注： 时序图说明了 SREN = 1 并且 BRGH = 0 时的同步主控模式。

表 14-8： 与同步主控接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
RCREG	EUSART 接收寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用 （读为 0）。同步主控接收不使用阴影单元。

14.4 EUSART 同步从动模式

将 CSRC （TXSTA<7>）位清零可进入同步从动模式。此模式与同步主控模式的区别在于移位时钟由 CK 引脚上的外部时钟提供（主控模式中由内部时钟提供）。这使得器件能在任何低功耗模式下发送或接收数据。

14.4.1 EUSART 同步从动发送

除了休眠模式以外，同步主控、从动模式的工作方式是完全相同的。

如果向 TXREG 写入两个字，然后执行 SLEEP 指令，则将发生以下事件：

a) 第一个字立即传送到 TSR 寄存器进行发送。
b) 第二个字仍保留在 TXREG 寄存器中。
c) 此时不会将标志位 TXIF 置 1。
d) 当第一个字移出 TSR 后，TXREG 寄存器将把第二个字传送给 TSR，同时将标志位 TXIF 置 1。
e) 如果中断允许位 TXIE 已置 1，中断将把器件从休眠状态唤醒。如果允许全局中断，程序则会跳转到中断向量处执行。

设置同步从动发送操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1、CSRC 位清零，使能同步从动串行端口。
2. 将 CREN 和 SREN 位清零。
3. 若需要中断，将中断允许位 TXIE 置 1。
4. 若需要发送 9 位数据，将 TX9 位置 1。
5. 将使能位 TXEN 置 1 使能发送。
6. 如果选择发送 9 位数据，应该将第 9 位数据装入 TX9D 位。
7. 将数据装入 TXREG 寄存器，启动发送。
8. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

表 14-9： 与同步从动发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
TXREG	EUSART 发送寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用 （读为 0）。同步从动发送不使用阴影单元。

14.4.2 EUSART 同步从动接收

除了休眠模式、空闲模式以及在从动模式下忽略 SREN 位以外，同步主控和从动模式的工作方式完全相同。

如果在进入休眠或任何空闲模式前将 CREN 位置 1，使能接收，那么在低功耗模式下可以接收到一个数据字。接收到该字后，RSR 寄存器将把数据传输到 RCREG 寄存器。如果中断允许位 RCIE 已置 1，产生的中断将把器件从低功耗模式唤醒。如果允许全局中断，程序则会跳转到中断向量处执行。

设置同步从动接收操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零使能同步从动串行端口。
2. 若需要中断，将中断允许位 RCIE 置 1。
3. 若需要接收 9 位数据，将 RX9 位置 1。
4. 将使能位 CREN 置 1，使能接收。
5. 当接收完成时，RCIF 标志位将置 1。如果中断允许位 RCIE 已置 1，还将产生一个中断。
6. 读 RCSTA 寄存器获取第 9 位数据（如果已使能），并判断在接收过程中是否发生了错误。
7. 通过读RCREG寄存器来读取接收到的8位数据。
8. 如果发生错误，将 CREN 位清零以清除错误。
9. 若想使用中断，请确保将 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

表 14-10： 与同步从动接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	42
RCREG	EUSART 接收寄存器								42
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	42
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	42
SPBRGH	EUSART 波特率发生器寄存器的高字节								42
SPBRG	EUSART 波特率发生器寄存器的低字节								42

图注： — = 未用（读为 0）。同步从动接收不使用阴影单元。

注：

15.0 10 位模数转换器 （A/D）模块

18/20/28 引脚器件的模数 （A/D）转换器模块具有 4 路输入。PIC18F1230/1330 器件的模数转换器模块能将一个模拟输入信号转换成相应的 10 位数字信号。

此模块有 5 个寄存器：

- A/D 结果寄存器的高字节 （ADRESH）
- A/D 结果寄存器的低字节 （ADRESL）
- A/D 控制寄存器 0 （ADCON0）
- A/D 控制寄存器 1 （ADCON1）
- A/D 控制寄存器 2 （ADCON2）

如寄存器 15-1 所示，A/D 模块的工作由 ADCON0 寄存器控制。如寄存器 15-2 所示，端口引脚的功能由 ADCON1 寄存器配置。如寄存器 15-3 所示，ADCON2 寄存器配置 A/D 时钟源，可编程的采集时间和输出结果的对齐方式。

寄存器 15-1： ADCON0：A/D 控制寄存器 0

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTEN	—	—	—	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **SEVTEN：**特殊事件触发器使能位
1 = 使能电源控制 PWM 模块的特殊事件触发器
0 = 禁止电源控制 PWM 模块的特殊事件触发器 （默认状态）

bit 6-4 未用：读为 0

bit 3-2 **CHS1:CHS0：**模拟通道选择位
00 = 通道 0 （AN0）
00 = 通道 1 （AN1）
00 = 通道 2 （AN2）
00 = 通道 3 （AN3）

bit 1 **GO/$\overline{\text{DONE}}$：**A/D 转换状态位
当 ADON = 1 时：
1 = A/D 转换正在进行
0 = A/D 空闲

bit 0 **ADON：**A/D 模块使能位
1 = 使能 A/D 转换器模块
0 = 禁止 A/D 转换器模块

寄存器 15-2： ADCON1：A/D 控制寄存器 1

U-0	U-0	U-0	R/W-0	R/W-0[1,2]	R/W[1]	R/W[1]	R/W[1]
—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-5 未用：读为 0

bit 4 **VCFG0**：参考电压配置位（VREF+ 电压源）
1 = A/D 模块的正参考电压是 VREF+
0 = A/D 模块的正参考电压是 AVDD

bit 3 **PCFG3**：RA6/AN3 的 A/D 端口配置位[1,2]
1 = 端口配置为 AN3
0 = 端口配置为 RA6

bit 2 **PCFG2**：RA4/AN2 的 A/D 端口配置位[1]
1 = 端口配置为 AN2
0 = 端口配置为 RA4

bit 1 **PCFG1**：RA1/AN1 的 A/D 端口配置位[1]
1 = 端口配置为 AN1
0 = 端口配置为 RA1

bit 0 **PCFG0**：RA0/AN0 的 A/D 端口配置位[1]
1 = 端口配置为 AN0
0 = 端口配置为 RA0

注 **1**： 复位后这些位被复位为 1，从而将端口配置为模拟输入端口。
2： 如果引脚没有被配置为 RA6，那么此位不使用且读为 0。

寄存器 15-3： ADCON2：A/D 控制寄存器 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **ADFM**：A/D 结果格式选择位
1 = 右对齐
0 = 左对齐

bit 6 未用：读为 0

bit 5-3 **ACQT2:ACQT0**：A/D 采集时间选择位
111 = 20 个 TAD
110 = 16 个 TAD
101 = 12 个 TAD
100 = 8 个 TAD
011 = 6 个 TAD
010 = 4 个 TAD
001 = 2 个 TAD
000 = 0 个 TAD(1)

bit 2-0 **ADCS2:ADCS0**：A/D 转换时钟选择位
111 = FRC （由 A/D RC 振荡器产生时钟信号）(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC （由 A/D RC 振荡器产生时钟信号）(1)
110 = FOSC/32
001 = FOSC/8
000 = FOSC/2

注 1： 如果选择了 A/D FRC 时钟源，在 A/D 时钟启动之前会添加一个 TCY （指令周期）的延时。这允许在开始转换之前执行 SLEEP 指令。

模拟参考电压可通过软件来选择，该参考电压可以是器件的正电源电压 （VDD）或 RA4/T0CKI/AN2/VREF+ 引脚上的电压。

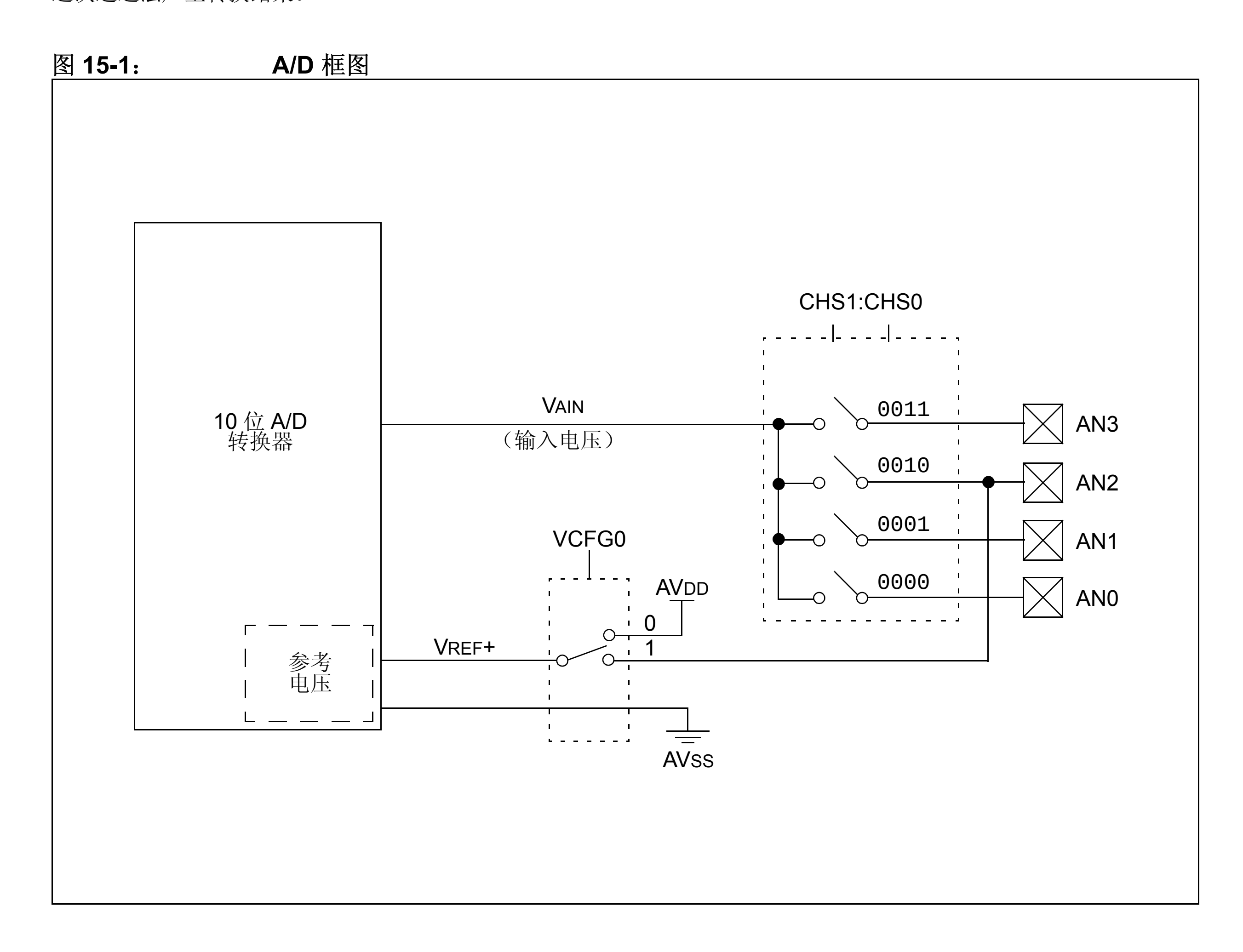
A/D 转换器具备一个独特的特性，它可在器件处于休眠模式时正常工作。要使 A/D 转换器在休眠模式下运行，A/D 转换时钟必须来自于 A/D 模块内部的 RC 振荡器。

采样保持电路的输出是转换器的输入，A/D 转换器采用逐次逼近法产生转换结果。

器件复位强制所有寄存器进入复位状态，同时迫使 A/D 模块关闭并中止任何正在进行的转换。

可以将每个与 A/D 转换器相关的端口引脚配置为模拟输入或数字 I/O。ADRESH 和 ADRESL 寄存器保存 A/D 转换的结果。当 A/D 转换完成之后，转换结果被载入 ADRESH:ADRESL 寄存器对，GO/DONE 位（ADCON0 寄存器）被清零且 A/D 中断标志位 ADIF 置 1。图 15-1 给出了 A/D 模块的框图。

图 15-1： A/D 框图



 超前信息

上电复位时， ADRESH:ADRESL 寄存器中的值保持不变。上电复位后，ADRESH:ADRESL 寄存器中的值不确定。

在根据需要配置好 A/D 模块之后，必须在转换开始之前对选定的通道进行采集。必须将模拟输入通道对应的 TRIS 位选择为输入。请参见**第 15.2 节 "A/D 采集要求"**确定采集时间。当采集完成后，A/D 转换即可开始。可将采集时间编程设定为在 GO/DONE 位置 1 和实际转换启动之间。

在执行 A/D 转换时应该遵循以下步骤：

1. 配置 A/D 模块：
   - 配置模拟引脚、参考电压和数字 I/O （通过 ADCON1 寄存器）
   - 选择 A/D 输入通道 （通过 ADCON0 寄存器）
   - 选择 A/D 采集时间 （通过 ADCON2 寄存器）
   - 选择 A/D 转换时钟 （通过 ADCON2 寄存器）
   - 使能 A/D 模块 （通过 ADCON0 寄存器）
2. 需要时，配置 A/D 中断：
   - 将 ADIF 位清零
   - 将 ADIE 位置 1
   - 将 GIE 位置 1
3. 如果需要的话，等待所要求的采集时间。
4. 启动转换：
   - 将 GO/DONE 位置 1（通过 ADCON0 寄存器）
5. 等待 A/D 转换完成，通过以下两种方法之一可判断转换是否完成：
   - 查询 GO/DONE 位是否被清零

   或
   - 等待 A/D 转换中断
6. 读取 A/D 结果寄存器 （ADRESH:ADRESL）；需要时将 ADIF 位清零。
7. 如需再次进行 A/D 转换，请根据要求返回步骤 1 或步骤 2。每位的 A/D 转换时间定义为 TAD。在下一次采集开始前至少需要等待 2 个 TAD。

图 15-2： A/D 传递函数

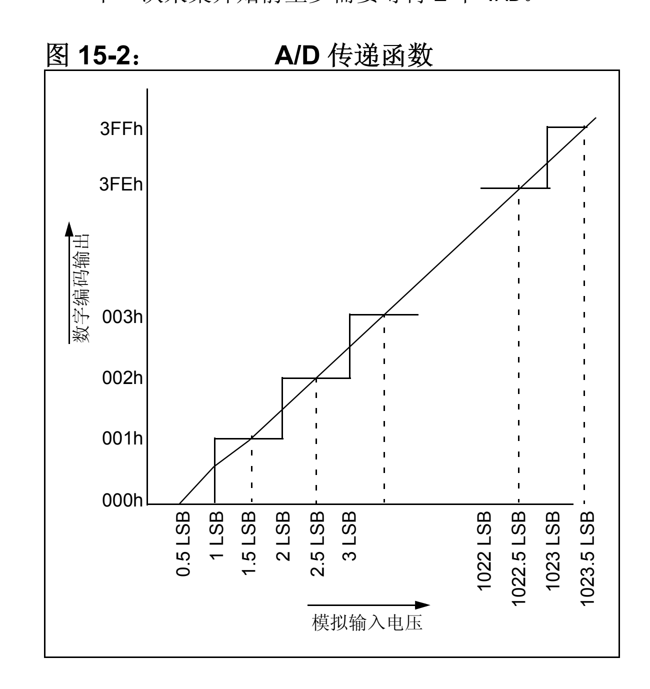
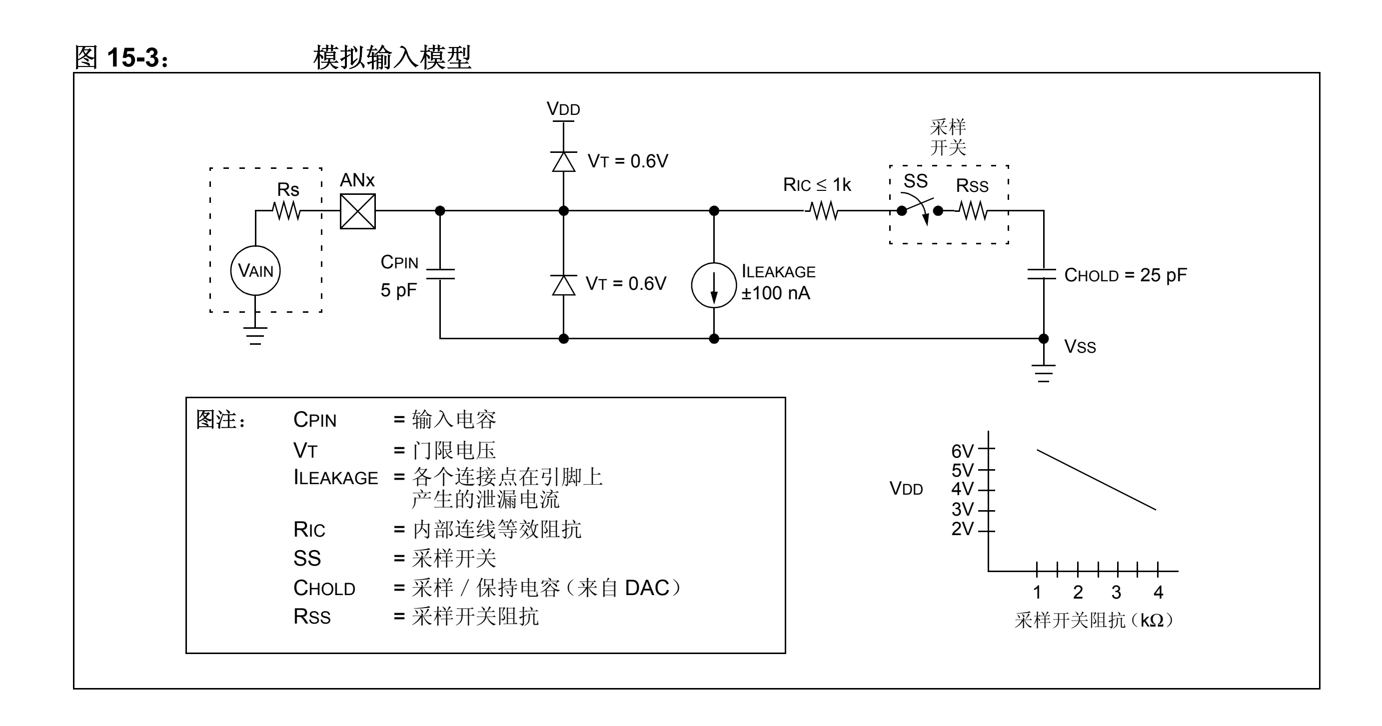


图 15-3： 模拟输入模型



15.1 触发 A/D 转换

通过将 GO/$\overline{\text{DONE}}$ 位置 1 可触发 A/D 转换。该位可由编程人员手工置 1，也可通过将 ADCON0 中的 SEVTEN 位置 1 来实现。当 SEVTEN 位置 1 后，来自电源控制 PWM 模块的特殊事件触发信号将触发 A/D 转换。更多信息请参见**第 13.14 节 "PWM 特殊事件触发器"**。

15.2 A/D 采集要求

为了使 A/D 转换器达到规定精度，必须让充电保持电容（CHOLD）充满至输入通道的电平。图 15-3 显示了模拟输入的模型。源阻抗（RS）和内部采样开关阻抗（RSS）直接影响给电容 CHOLD 充电所需要的时间。采样开关阻抗（RSS）随器件电压（VDD）的变化而变化。源阻抗影响模拟输入的失调电压（由于引脚泄漏电流）。**建议模拟信号源的最大阻抗为 2.5 kΩ**。在选择（改变）模拟输入通道后，必须对通道进行采样才能开始转换，采样时间必须大于最小采集时间。

注：	当开始转换时，应断开保持电容与输入引脚的连接。

可使用公式 15-1 来计算最小采集时间。该公式假设使用的量化误差为 1/2 LSb（A/D 转换需要 1024 步）。1/2 LSb 的误差是 A/D 模块达到规定分辨率所能允许的最大误差。

公式 15-3 显示了所需的最小采集时间 TACQ 的计算过程。计算结果基于以下对应用系统的假设：

CHOLD	=	25 pF
Rs	=	2.5 kΩ
转换误差	≤	1/2 LSb
VDD	=	5V → RSS = 2 kΩ
温度	=	85°C（系统最大值）

公式 15-1： 采集时间

$$T_{ACQ} = \text{放大器稳定时间} + \text{保持电容充电时间} + \text{温度系数}$$
$$= T_{AMP} + T_C + T_{COFF}$$

公式 15-2： **A/D 最小充电时间**

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

或

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)$$

公式 15-3： 计算所需的最小采集时间

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{AMP} = 0.2\ \mu s$$

$$T_{COFF} = (Temp - 25°C)(0.02\ \mu s/°C)$$
$$= (85°C - 25°C)(0.02\ \mu s/°C)$$
$$= 1.2\ \mu s$$

只有在温度 > 25°C 时才需要温度系数。当温度低于 25°C 时，TCOFF = 0 ms。

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2047)$$
$$= -(25\ pF)(1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega)\ln(0.0004883)$$
$$= 1.05\ \mu s$$

$$T_{ACQ} = 0.2\ \mu s + 1\ \mu s + 1.2\ \mu s$$
$$= 2.4\ \mu s$$

 超前信息

15.3 选择和配置采集时间

ADCON2寄存器允许用户选择采集时间，该采集时间发生在每次 GO/DONE 位置 1 后。它还为用户提供了使用系统自定义采集时间的选项。

可以使用 ACQT2:ACQT0 位（ADCON2<5:3>）设置采集时间，采集时间的范围是 2 到 20 TAD。当 GO/DONE 位置 1 时，A/D 模块继续对输入进行采样，采样时间为所选择的采集时间，然后自动开始转换。因为采集时间已被编程，就不需要在选择通道以后等待一个采集时间，再将 GO/DONE 位置 1。

若 ACQT2:ACQT0 = `000`，则表示选择手动采集。当 GO/DONE 位置 1 时，采样停止，转换开始。用户需确保在选择所需要的输入通道和 GO/DONE 置 1 之间经过了必需的采集时间。此选项也是 ACQT2:ACQT0 位的默认复位状态，并且与不提供可编程采集时间的器件兼容。

在这两种情况下，当转换完成时，GO/DONE位被清零、ADIF 标志位被置 1 且 A/D 再次开始对当前选择的通道进行采样。如果编程了采集时间，那么将不会有标志显示采集时间是否结束和转换是否开始。

15.4 选择 A/D 转换时钟

每位的 A/D 转换时间被定义为 TAD。每完成一次 10 位 A/D 转换需要 11 个 TAD 。可用软件选择 A/D 转换的时钟源。 TAD 有以下 7 种可能的选择：

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- 内部 RC 振荡器

要进行正确的 A/D 转换， A/D 转换时钟 （TAD）必须尽可能小，但它必须大于最小 TAD （欲知更多信息，见参数 130）。

表 15-1 显示了器件在不同的工作频率下和选择不同的 A/D 时钟源时得到的 TAD。

表 15-1： 不同器件工作频率下的 TAD

AD 时钟源 （TAD）		最高器件频率	
工作模式	ADCS2:ADCS0	PIC18F1230/1330	PIC18LF1230/1330(4)
2 TOSC	`000`	2.86 MHz	1.43 kHz
4 TOSC	`100`	5.71 MHz	2.86 MHz
8 TOSC	`001`	11.43 MHz	5.72 MHz
16 TOSC	`101`	22.86 MHz	11.43 MHz
32 TOSC	`010`	40.0 MHz	22.86 MHz
64 TOSC	`110`	40.0 MHz	22.86 MHz
RC(3)	`x11`	1.00 MHz(1)	1.00 MHz(2)

注 **1：** RC 时钟源的典型 TAD 时间是 1.2 μs。

2： RC 时钟源的典型 TAD 时间是 2.5 μs。

3： 当器件工作频率高于 1 MHz 时，整个转换过程必须在休眠模式下进行，否则 A/D 转换精度可能超出规范允许的范围。

4： 仅低功耗器件 （PIC18LF1230/1330）。

15.5 在功耗管理模式下的工作

在功耗管理模式中，自动采集时间和 A/D 转换时钟的选择在一定程度上可由时钟源和频率决定。

如果要在器件处于功耗管理模式时进行 A/D 转换，ADCON2 中的 ACQT2:ACQT0 和 ADCS2:ADCS0 位就应该根据要在功耗管理模式中使用的时钟进行更新。在进入该模式之后，可以开始 A/D 采集或转换。采集或转换开始以后，器件仍应继续使用相同的时钟源直到转换完成。

如果需要的话，在转换期间也可以将器件置于相应的功耗管理空闲模式。如果器件时钟频率小于 1 MHz，就应该选择 A/D RC 时钟源。

在休眠模式下工作需要选择 A/D FRC 时钟。如果将 ACQT2:ACQT0 位设置为 `000` 并启动转换，转换将延迟一个指令周期以允许执行 `SLEEP` 指令并进入休眠模式。在转换开始前，必须首先将 IDLEN 位（OSCCON<7>）清零。

15.6 配置模拟端口引脚

ADCON1 和 TRISA 寄存器用于配置 A/D 端口引脚的操作。若希望端口引脚为模拟输入，则必须将相应的 TRIS 位置 1 （输入）。如果 TRIS 位被清零 （输出），则将对该引脚的数字输出电平（VOH 或 VOL）进行相应的转换。

A/D 转换操作与 CHS3:CHS0 位以及 TRIS 位的状态无关。

注	
1：	读取端口寄存器时，所有配置为模拟输入通道的引脚均读为 0（低电平）。配置为数字输入的引脚将对模拟输入信号进行转换。引脚上的模拟电平将被正确转换为数字电平。
2：	定义为数字输入引脚上的模拟电平可能会导致数字输入缓冲器消耗的电流超出器件规范。

15.7 A/D 转换

图 15-4 显示了在 GO/DONE 位置 1 且 ACQT2:ACQT0 位被清零后 A/D 转换器的工作状态。转换在下一条指令执行之后开始，以允许器件在转换开始之前进入休眠模式。

图 15-5 显示了在 GO/DONE 位置 1 且 ACQT2:ACQT0 位被设置为 `010`（即在转换开始之前选择了 4 TAD 的采集时间）后 A/D 转换器的工作状态。

在转换期间将 GO/DONE 位清零将中止当前的转换。不会用部分完成的 A/D 转换结果更新 A/D 结果寄存器对。这意味着 ADRESH:ADRESL 寄存器对仍将保持上一次转换完成后的值 （或上一次写入 ADRESH:ADRESL 寄存器的值）。

在 A/D 转换完成或停止以后，需要等待 2 个 TAD 才能开始下一次采集。等待结束后将自动开始对所选通道进行采集。

注：	不应在启动 A/D 模块的同一指令中将 GO/DONE 位置 1。

15.8 放电

放电过程用于对电容阵列进行初始化。在每次采样前，均需要对该阵列放电。这样做有助于优化单位增益放大器，因为每次需要重新为电容阵列充电，而不是根据以前测量的值进行充放电。

图 15-4： A/D 转换 TAD 周期 （ACQT<2:0> = 000， TACQ = 0）

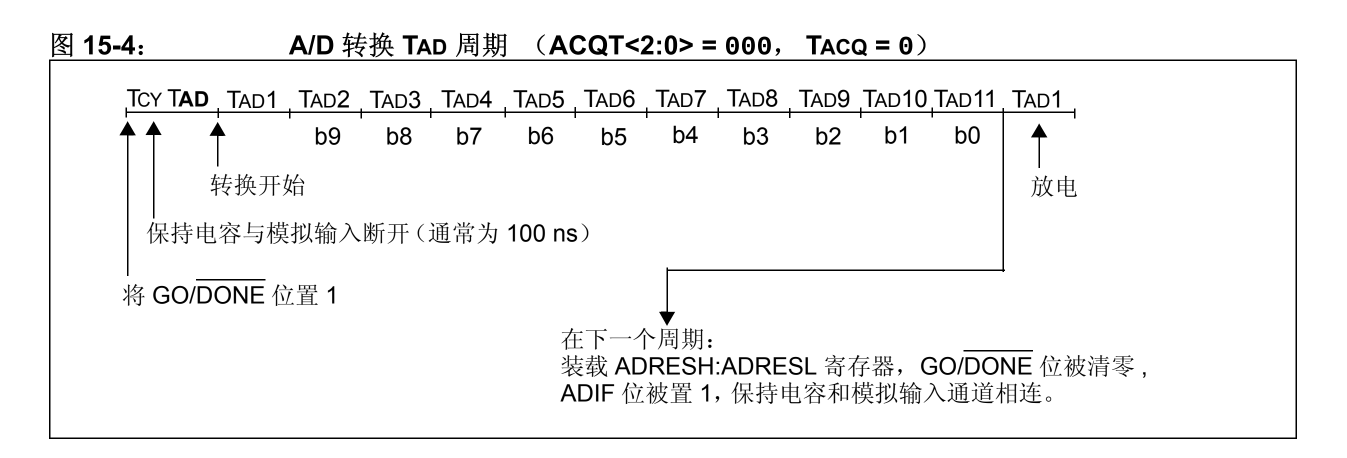


图 15-5： A/D 转换 TAD 周期 （ACQT<2:0> = 010，TACQ = 4 TAD）

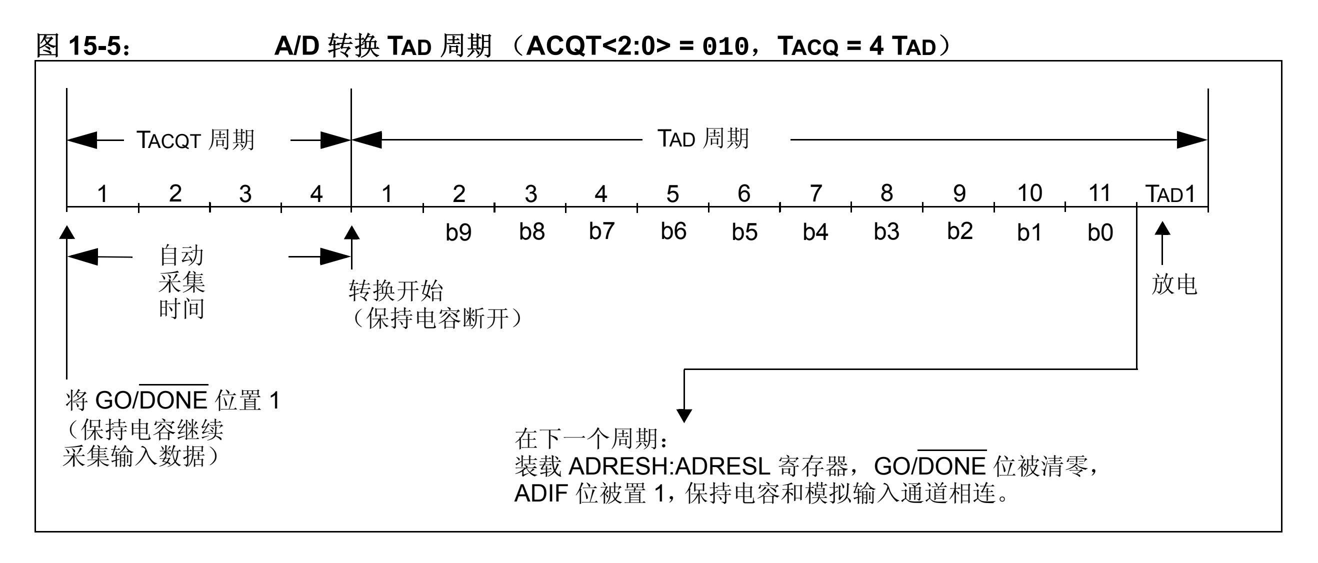


表 15-2： 与 A/D 模块的操作相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
ADRESH	A/D 结果寄存器的高字节								42
ADRESL	A/D 结果寄存器的低字节								42
ADCON0	SEVTEN	—	—	—	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	42
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	42
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	42
PORTA	RA7(1)	RA6(1)	RA5(2)	RA4	RA3	RA2	RA1	RA0	44
TRISA	TRISA7(1)	TRISA6(1)	PORTA 数据方向控制寄存器						43

图注： — = 未用 （读为 0）。 A/D 转换未使用阴影单元。

注 1： PORTA<7:6> 及其相关数据方向位可根据不同的主振荡器模式被单独配置为端口引脚。禁止时，这些位读为 0。

2： 只有当禁止主清零复位 （MCLRE 配置位 = 0）时， RA5 位才可用； 否则 RA5 读为 0。此位是只读位。

16.0 比较器模块

模拟比较器模块包含三个比较器。可以选择与 RA0、RB2 和 RB3 引脚复用的模拟输入及片上参考电压 （见第 **17.0** 节 **"比较器参考电压模块"**）作为比较器的输入。数字输出不能通过引脚电平获取，只能通过控制寄存器 CMCON （寄存器 16-1）读取。CMCON 也用于选择比较器的输入。

寄存器 16-1： CMCON：比较器控制寄存器

R-0	R-0	R-0	U-0	U-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **C2OUT**：比较器 2 输出位
1 = C2 VIN+ > C2 VIN- （CVREF）
0 = C2 VIN+ < C2 VIN- （CVREF）

bit 6 **C1OUT**：比较器 1 输出位
1 = C1 VIN+ > C1 VIN- （CVREF）
0 = C1 VIN+ < C1 VIN- （CVREF）

bit 5 **C1OUT**：比较器 0 输出位
1 = C0 VIN+ > C0 VIN- （CVREF）
0 = C0 VIN+ < C0 VIN- （CVREF）

bit 4-3 未用位：读为 0

bit 2 **CMEN2**：比较器 2 使能位
1 = 使能比较器 2
0 = 禁止比较器 2

bit 1 **CMEN1**：比较器 1 使能位
1 = 使能比较器 1
0 = 禁止比较器 1

bit 0 **CMEN0**：比较器 0 使能位
1 = 使能比较器 0
0 = 禁止比较器 0

16.1 比较器配置

对于每个模拟比较器，在 CMCON 寄存器中均有一个称为 CMENx 的控制位。通过将 CMENx 位置 1，可以使能对应的比较器。如果改变比较器模式，由于存在特定的模式改变延迟（如第 **22.0 节 "电气规范"** 所示）比较器的输出电平可能会在此延迟期间无效。

注：	改变比较器工作模式时应禁止比较器中断，以免产生误中断。

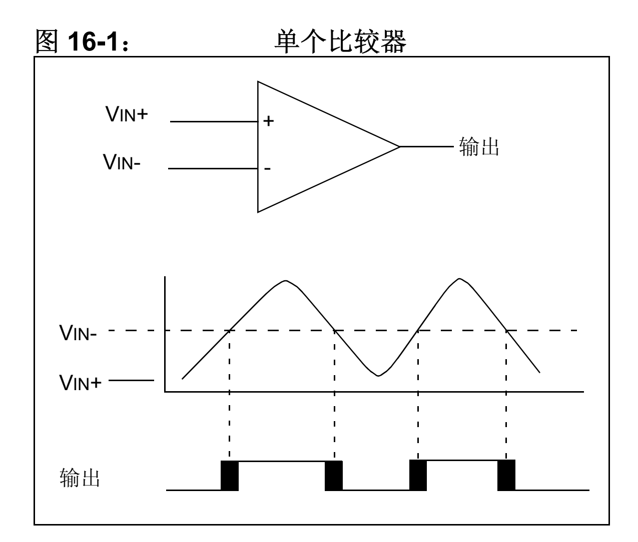
16.2 比较器工作原理

图16-1所示为单个比较器以及模拟输入电平和数字输出之间的关系。如果 VIN+（CMPx）上的模拟输入电平小于 VIN-（CVREF）上的模拟输入电平，那么比较器将输出数字低电平。当 VIN+（CMPx）上的模拟输入电平高于 VIN-（CVREF）上的模拟输入电平时，比较器输出数字高电平。图 16-1 中比较器输出的阴影部分表示因输入失调和响应时间所造成的不确定区域。

16.3 比较器参考电压

在此比较器模块中，使用的是内部参考电压（见第 **17.0 节 "比较器参考电压模块"**）。

图 16-1： 单个比较器



16.4 比较器的响应时间

响应时间是指从选定一个新的参考电压或输入源到比较器输出达到一个有效电平的最短时间。如果内部参考电压发生了改变，在使用比较器的输出时必须考虑到内部参考电压的最大延时。否则，应该使用比较器的最大延时（见第 **22.0 节 "电气规范"**）。

16.5 比较器输出

通过读 CMCON 寄存器的 CxOUT 位，可以得到比较器的输出，这些位是只读的。每个比较器的不确定区的大小与输入失调电压和响应时间（在规范中给出）有关。

注	1：	当读取 PORT 寄存器时，所有配置为模拟输入的引脚都读为 0。配置为数字输入的引脚将根据施密特触发器输入规范转换模拟输入信号。
	2：	对被定义为数字输入的任何引脚施加模拟电平，均可能会使输入缓冲器的电流消耗超过规定值。

16.6 比较器中断

一旦比较器的输出值发生变化就会将相应比较器的中断标志位置 1。软件需要从 CMCON<7:5> 读取数据来保持输出位的状态信息，以判断实际发生的变化。CMPxIF （PIR1<3:1>）位是比较器中断标志位。CMPxIF 位必须通过清零复位。因为也可以向 CMCON 寄存器写入 1，所以可以模拟中断的发生。

必须将 CMPxIE 位（PIE1<3>）和 PEIE 位（INTCON<6>）置 1 允许相应比较器的中断。此外，还必须将 GIE （INTCON<7>）位置 1。只要这些位中有一位清零，虽然当有中断条件产生时 CMPxIF 位仍会置 1，但中断却是被禁止的。

注：	当执行读操作时 （Q2 周期开始时），如果 CMCON 寄存器 （C2OUT、C1OUT 或 C0OUT）发生变化，那么 CMPxIF （PIR1 寄存器）中断标志位可能不会被置 1。

用户可用以下方式在中断服务程序中清除该中断：

a) 对CMCON寄存器的任何读或写操作都将终止不匹配条件。
b) 将 CMPxIF 标志位清零。

不匹配条件会一直不断地将 CMPxIF 标志位置 1。读 CMCON 寄存器将结束不匹配状态，并允许将 CMPxIF 标志位清零。

 超前信息

16.7 比较器在休眠模式下的工作原理

当比较器处于活动状态而器件处于休眠模式时，比较器仍可正常工作并可使用比较器中断（如果允许的话）。在允许中断时，中断会把器件从休眠模式唤醒。每个比较器工作时都会消耗额外的电流，如比较器规范中所示。若要把休眠状态下的功耗减少到最小，可在进入休眠模式前关闭比较器模块（CM2:CM0 = 000）。器件从休眠模式唤醒时，CMCON 寄存器的内容不受影响。

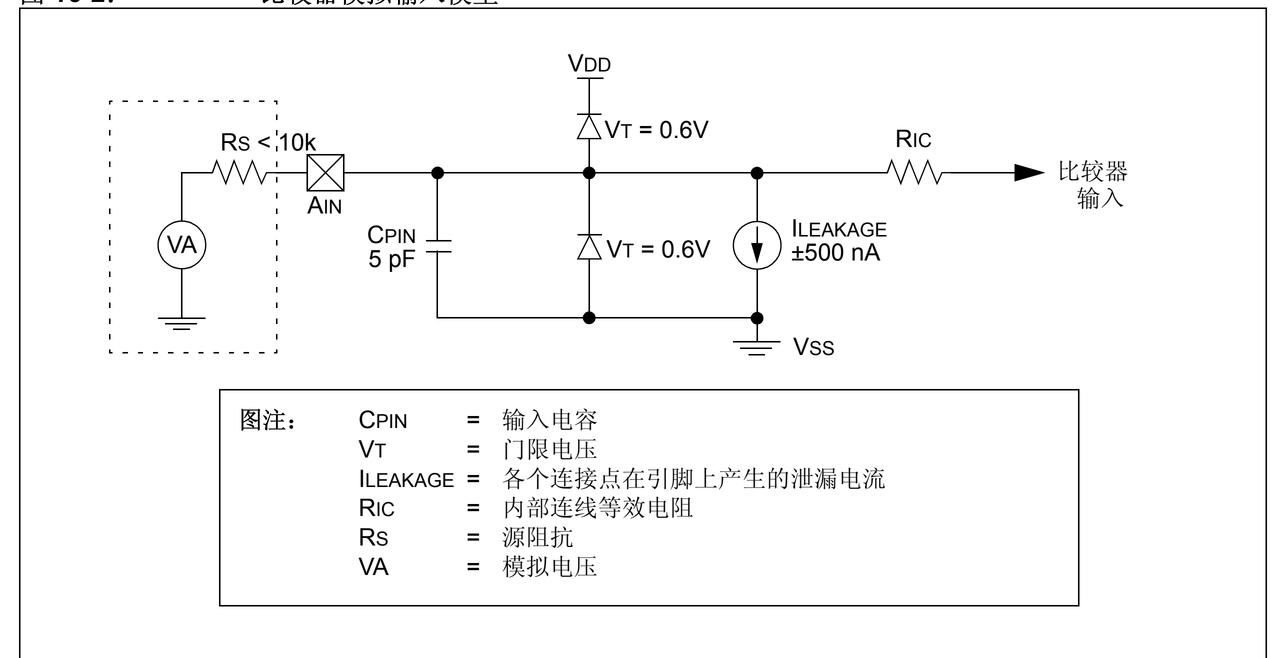
16.8 复位的影响

器件复位强制 CMCON 寄存器进入复位状态，导致比较器模块被关闭（CMEN2:CMEN0 = 000）。

16.9 模拟输入连接注意事项

图 16-2 所示为一个简化的模拟输入电路。由于模拟引脚和数字输出端相连，因而它们与 VDD 和 VSS 之间加有反向偏置的二极管，从而将模拟输入电压限制在 VSS 和 VDD 之间。一旦输入电压超出该范围 0.6V 以上，就会有一个二极管正偏从而使输入电压被钳位。模拟信号源的最大阻抗值推荐为 10 kΩ。任何连接到模拟输入引脚的外部元件（如电容和齐纳二极管等）的泄漏电流应该极小。

图 16-2： 比较器模拟输入模型



图注：		
CPIN	=	输入电容
VT	=	门限电压
ILEAKAGE	=	各个连接点在引脚上产生的泄漏电流
RIC	=	内部连线等效电阻
RS	=	源阻抗
VA	=	模拟电压

表 16-1： 与比较器模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	42
CVRCON	CVREN	—	—	CVRSS	CVR3	CVR2	CVR1	CVR0	42
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	43
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	43
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	43
PORTA	RA7[1]	RA6[1]	RA5[2]	RA4	RA3	RA2	RA1	RA0	44
LATA	LATA7[1]	LATA6[1]	PORTA 数据锁存器 （读取和写入数据锁存器）						43
TRISA	TRISA7[1]	TRISA6[1]	PORTA 数据方向控制寄存器						43
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	44
LATB	PORTB 数据锁存器 （读取和写入数据锁存器）								43
TRISB	PORTB 数据方向控制寄存器								43

图注： — = 未用 （读为 0）。比较器模块不使用阴影单元。

注 1： 根据各种主振荡器模式， PORTA<7:6> 及其方向和锁存位可被单独配置为端口引脚。这些位在禁止时读为 0。

2： RA5 位只有在禁止主清零复位时 （MCLRE 配置位 = 0）才可用；否则， RA5 读为 0。该位是只读位。

 超前信息

17.0 比较器参考电压模块

比较器参考电压模块是一个 16 阶的梯形电阻网络，提供多个参考电压以供选择。其目的是为模拟比较器提供参考电压。

图 17-1 给出了该模块的框图。梯形电阻分成两组可提供两种量程范围的 CVREF 值，并且还具有断电功能以在不使用参考电压时降低功耗。模块的参考电源由器件 VDD/VSS 或外部参考电压提供。

17.1 配置比较器参考电压

参考电压模块由 CVRCON 寄存器（寄存器 17-1）控制，可提供两种范围的输出电压，每种范围都具有 16 种不同的电平。CVRR 位（CVRCON<5>）选择要用的电压范围。这两种范围的主要区别在于由 CVREF 选择位（CVR3:CVR0）选定的步长不同，其中一个范围具有更高的分辨率。下面是计算比较器参考电压输出值的公式：

如果 CVRR = 1：
CVREF = ((CVR3:CVR0)/24) x CVRSRC

如果 CVRR = 0：
CVREF = (CVRSRC x 1/4) + (((CVR3:CVR0)/32) x CVRSRC)

比较器参考电压模块电源可以来自 AVDD 和 AVSS，或者与 RA2 和 AVSS 复用的外部 VREF+。CVRSS 位（CVRCON<4>）用于选择电压源。

在改变 CVREF 输出值时，必须考虑比较器参考电压的稳定时间（见**第 22.0 节“电气规范”**中的表 22-3）。

寄存器 17-1： CVRCON：比较器参考电压控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7 **CVREN**：比较器参考电压使能位
1 = CVREF 电路上电
0 = CVREF 电路断电

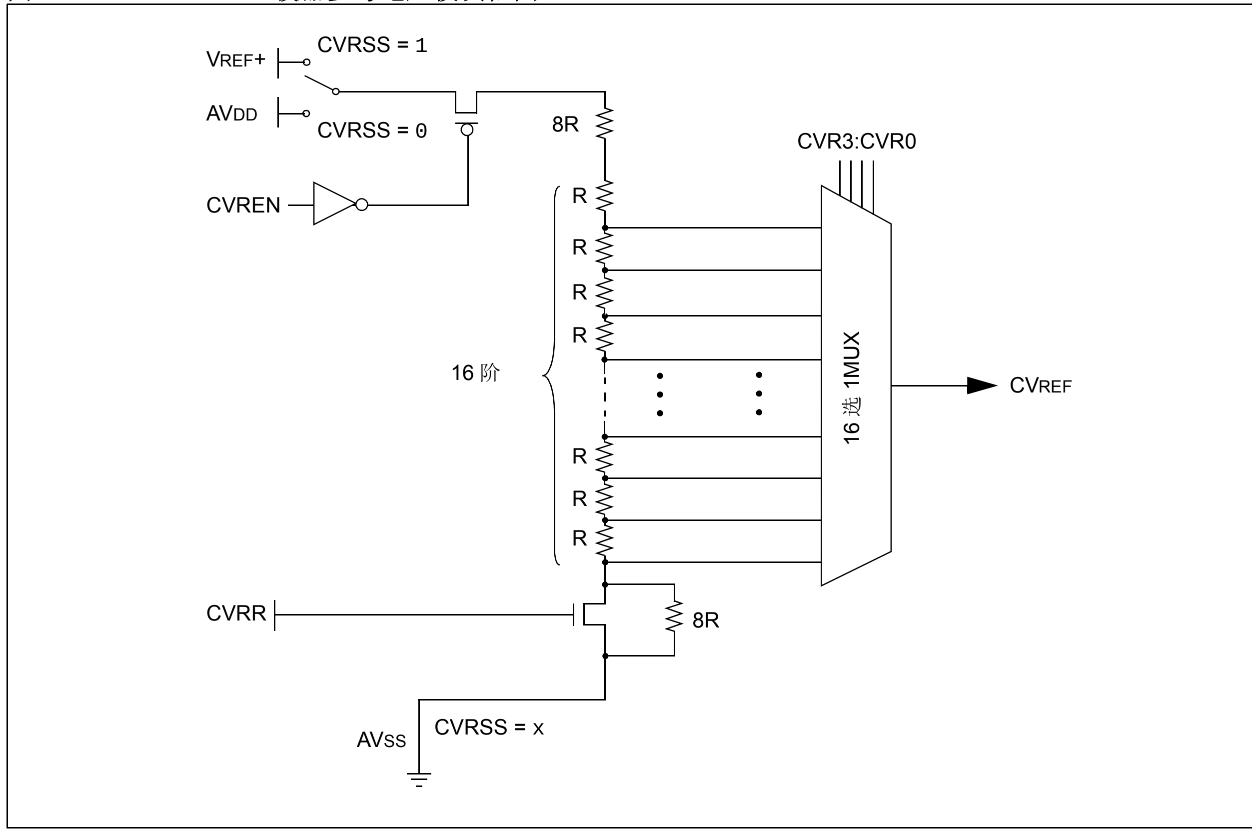
bit 6 未用：读为 0

bit 5 **CVRR**：比较器 VREF 量程范围选择位
1 = 0 到 0.667 CVRSRC，步长为 CVRSRC/24 （低电压范围）
0 = 0.25 CVRSRC 到 0.75 CVRSRC，步长为 CVRSRC/32 （高电压范围）

bit 4 **CVRSS**：比较器 VREF 源选择位
1 = 比较器参考电压源， CVRSRC = (VREF+) – (AVSS)
0 = 比较器参考电压源， CVRSRC = AVDD – AVSS

bit 3-0 **CVR3:CVR0**：比较器 VREF 值选择位 （0 ≤ (CVR3:CVR0) ≤ 15）
当 CVRR = 1 时：
CVREF = ((CVR3:CVR0)/24) • (CVRSRC)
当 CVRR = 0 时：
CVREF = (CVRSRC/4) + ((CVR3:CVR0)/32) • (CVRSRC)

图 17-1：　　比较器参考电压模块框图



17.2　参考电压精度 / 误差

由于模块结构的原因，模块无法输出满量程参考电压。梯形电阻网络中顶端和底部的晶体管 （图 17-1）使 CVREF 无法达到参考电压源的满幅值。参考电压来自于参考电压源；因此，CVREF 输出电平会随参考电压源一起波动。**第 22.0 节 “电气规范”**中可找到测试所得的参考电压绝对精度值。

17.3　休眠期间的操作

当中断或看门狗定时器超时唤醒器件时， CVRCON 寄存器内容不受影响。为了最大限度降低休眠模式下的电流消耗，应禁止参考电压模块。

17.4　复位的影响

器件复位通过清零 CVREN （CVRCON<7>）禁止参考电压模块，通过清零 CVRR （CVRCON<5>）位，选择高电压范围。同时 CVR 值选择位也被清零。

表 17-1：　　与比较器参考电压模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	42
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	42

图注：　比较器参考电压模块不使用阴影单元。

18.0 低电压检测 （LVD）

PIC18F1230/1330 器件有一个低电压检测 （LVD）模块。该模块是一个可编程电路，它允许用户指定器件的电压跳变点。如果器件电压相对于跳变点发生了偏移，就会将中断标志位置 1。如果允许中断，程序就会跳转到中断向量地址处执行，由软件响应该中断。

低电压检测控制寄存器 （寄存器 18-1）全面控制 LVD 模块的工作。这就允许用户通过软件控制将该电路 “关闭”，从而使器件的电流消耗降至最低。

图 18-1 所示为 LVD 模块的框图。

寄存器 18-1： LVDCON：低电压检测控制寄存器

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3[(1)]	LVDL2[(1)]	LVDL1[(1)]	LVDL0[(1)]
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-6 未用：读为 0

bit 5 **IRVST**：内部参考电压稳定标志位
1 = 表示电压检测逻辑将在指定的电压跳变点产生中断标志
0 = 表示电压检测逻辑将不会在指定的电压跳变点产生中断标志，并且不允许 LVD 中断

bit 4 **LVDEN**：低电压检测电源使能位
1 = 使能 LVD
0 = 禁止 LVD

bit 3-0 **LVDL3:LVDL0**：低电压检测门限值位[(1)]
1111 = 保留
1110 = 最大设置
.
.
.
0000 = 最小设置

注 1： 相应规范请参见**第 22.0 节 “电气规范”**中的表 22-4。

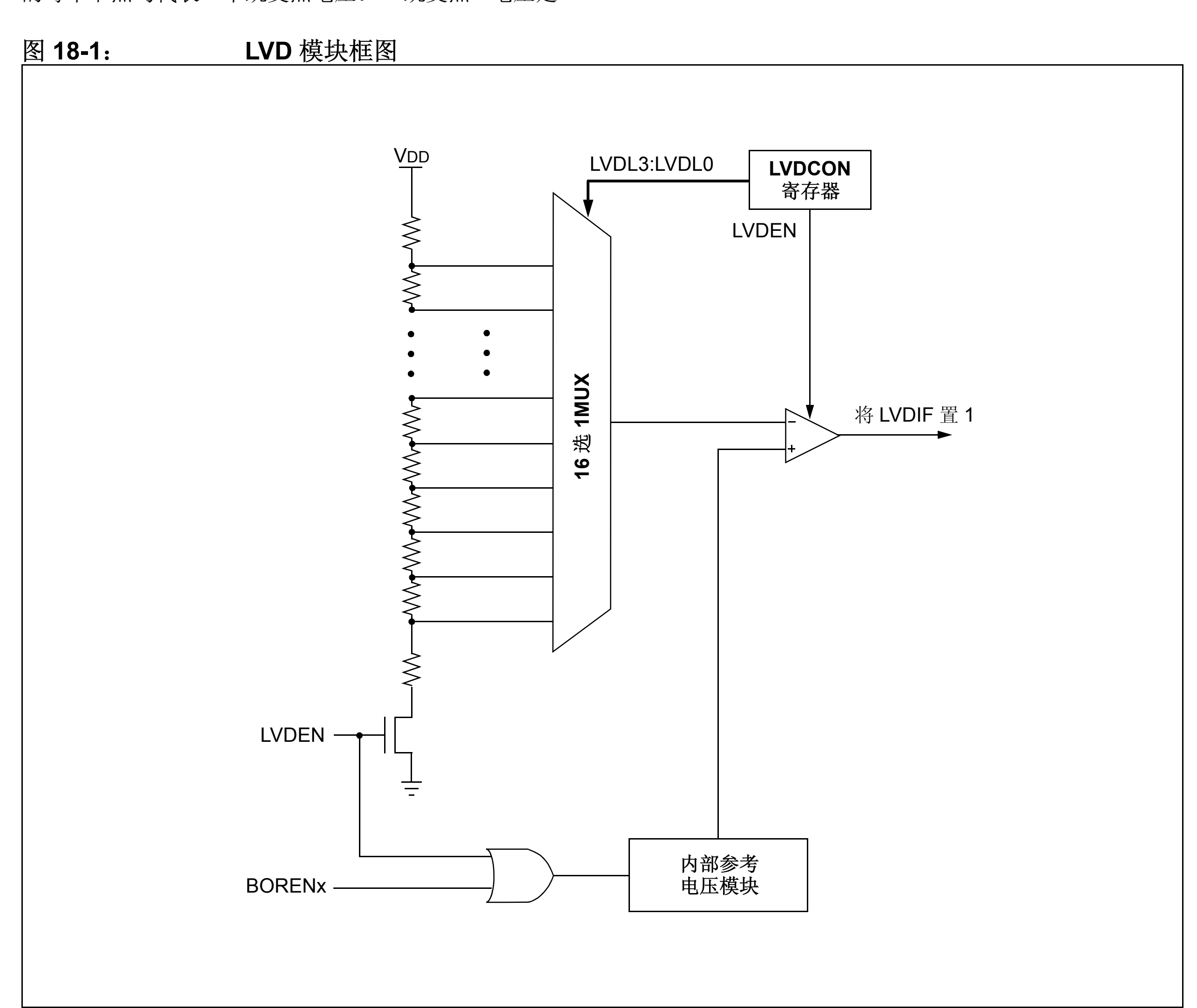
通过将 LVDEN 位置 1 使能该模块。每次使能 LVD 模块时，电路需要一段时间才能稳定下来。IRVST 位是一个只读位，用来表明电路是否稳定。仅当该电路稳定且 IRVST 位置 1 时，该模块才能产生中断。

18.1 工作原理

当使能 LVD 模块时，比较器使用内部产生的参考电压作为设置点。设置点与跳变点作比较，其中电阻分压器中的每个节点均代表一个跳变点电压。“跳变点”电压是器件在检测到低电压事件时的电平，它取决于该模块的配置。当供电电压等于跳变点时，电阻阵列分压值等于由参考电压模块产生的内部参考电压。然后比较器通过将 LVDIF 位置 1 产生一个中断信号。

可用软件编程指定跳变点电压为 15 个值中的任何一个。通过对 LVDL3:LVDL0 位 （LVDCON<3:0>）编程可以选择跳变点。

图 18-1： LVD 模块框图



18.2 设置 LVD

要设置 LVD 模块，需要遵循以下步骤：

1. 将 LVDEN 位（LVDCON<4>）清零禁止该模块。
2. 将值写入 LVDL3:LVDL0 位，选择所需的 LVD 跳变点。
3. 将 LVDEN 位置 1 使能 LVD 模块。
4. 将 LVD 中断标志位 （PIR2<2>）清零，该位可能从上次中断起一直保持置 1。
5. 如果需要中断，通过将LVDIE和GIE位（PIE2<2>和INTCON<7>）置1允许LVD中断。直到IRVST位置 1 后才会产生中断。

18.3 电流消耗

使能了该模块就使能了 LVD 比较器和分压器，并将消耗静态电流。电气规范中的参数 D022B 给出了使能该模块时消耗的总电流。

HLVD 模块不需要一直工作，工作与否取决于具体的应用。要降低电流消耗，只需要在检测电压时，短时间地使能 LVD 电路，而在检测完成之后再禁止 LVD 模块。

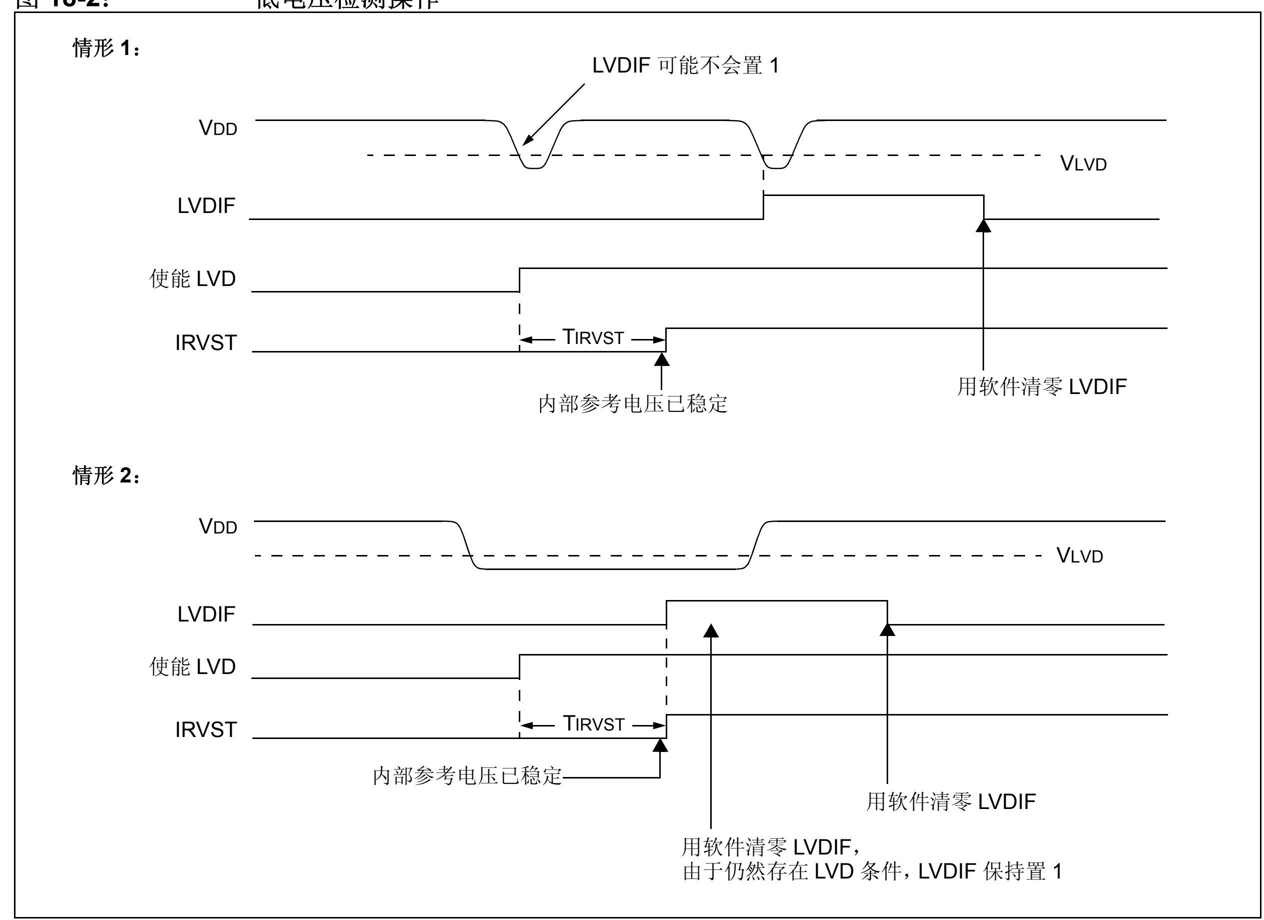
18.4 LVD 启动时间

电气规范中的参数 D420 给出了 LVD 模块的内部参考电压，该参考电压也可由其他内部电路使用，例如可编程欠压复位电路。如果禁止了LVD或其他使用参考电压的电路以降低器件的电流消耗，则参考电压电路将需要一段时间稳定下来以后才能可靠地检测低电压条件。启动时间 TIRVST 是一个独立于器件时钟速率的时间间隔，由电气规范参数 36 指定。

直到 TIRVST 结束并且参考电压达到稳定后，才会允许 LVD 中断。由于这个原因，在此时间间隔期间，短时间超出设置点的偏移可能不会被检测到 （见图 18-2）。



图 18-2： 低电压检测操作



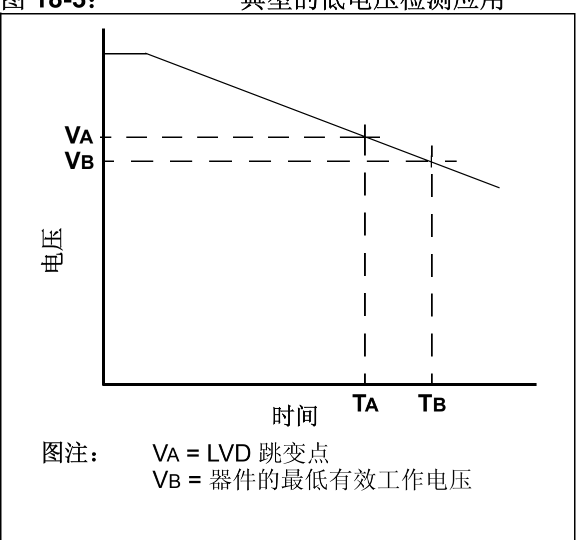
18.5 应用

在许多应用中，需要有检测电压跌落到某个特定门限值以下的功能。

对于一般的电池应用，图 18-3 显示了一个可能的电压曲线。器件电压会随时间逐渐下降。当器件电压降至电压 V_A 时，LVD 逻辑电路会在时间 T_A 处产生中断。中断将导致执行 ISR，从而使应用程序能在器件电压退出有效工作范围 （对应时间为 T_B）之前执行 “日常任务” 并安全关闭。因此， LVD 将会给应用提供一个时间窗 （表示为 T_A 和 T_B 的差）使应用程序能安全地退出。



图 18-3： 典型的低电压检测应用



18.6 休眠期间的操作

使能时，LVD 电路在休眠期间将继续工作。如果器件电压越过了跳变点，LVDIF 位将会置 1，并且将唤醒器件。如果已经允许了全局中断，程序将从中断向量地址处继续执行。

18.7 复位的影响

器件复位强制所有寄存器进入复位状态。这会强制关闭 LVD 模块。

表 18-1： 与低电压检测模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	42
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	41
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	43
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	43
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	43

图注： — = 未用 （读为 0）。 LVD 模块不使用阴影单元。

19.0 CPU 的特殊功能

PIC18F1230/1330 器件包含的功能旨在最大限度地提高系统可靠性，并通过减少外部元件把系统成本降到最低。这些功能包括：

- 振荡器选择
- 复位：
  - 上电复位 （POR）
  - 上电延时定时器 （PWRT）
  - 振荡器起振定时器 （OST）
  - 欠压复位 （BOR）
- 中断
- 看门狗定时器 （WDT）
- 故障保护时钟监视器 （Fail-Safe Clock Monitor，FSCM）
- 双速启动
- 代码保护
- ID 单元
- 在线串行编程

应根据具体应用对频率、功耗、精度和成本的要求配置振荡器。在**第 2.0 节 "振荡器配置"**中详细讨论了所有的选项。

在本数据手册的前面几章中已经完整地讨论了器件的复位和中断。

除了为复位提供了上电延时定时器和振荡器起振定时器之外，PIC18F1230/1330 器件还提供了一个看门狗定时器，该定时器可通过配置位永久使能或由软件控制（如果被配置为禁止）。

器件自带的 RC 振荡器还提供了故障保护时钟监视器（FSCM）和双速启动两个功能。 FSCM 对外设时钟进行后台监视，并在外设时钟发生故障时自动切换时钟源。双速启动使得代码几乎可在启动开始时立即执行，此时主时钟源正在完成自身的起振延时。

通过设置配置寄存器中相应的位可以使能和配置所有这些功能。

19.1 配置位

可以通过对配置位编程 （读为 `0`）或不编程 （读为 `1`）来选择不同的器件配置。这些配置位被映射到程序存储器中从 300000h 开始的单元中。

用户会注意到地址 300000h 超出了用户程序存储空间的范围。事实上，它属于配置存储空间 （300000h— 3FFFFFh），该空间仅能通过表读和表写指令进行访问。

对配置寄存器编程类似于对闪存存储器编程。EECON1 寄存器中的 WR 位可启动对配置寄存器的自定时写操作。在正常工作模式下， TBLPTR 指向配置寄存器，`TBLWT` 指令设置要用于写操作的地址和数据。将 WR 位置 1 可启动对配置寄存器的长写操作。每次往配置寄存器写入 1 个字节。要写入或擦除一配置单元，可用 `TBLWT` 指令分别对该单元写入 `1` 或 `0`。关于闪存编程的更多详情，请参见**第 6.5 节 "写入闪存程序存储器"**。

表 19-1： 配置位和器件 ID

寄存器名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	缺省 / 未编程值
300001h	CONFIG1H	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	00-- 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300004h	CONFIG3L	—	—	—	—	HPOL	LPOL	PWMPIN	—	---- 111-
300005h	CONFIG3H	MCLRE	—	—	—	T1OSCMX	—	—	FLTAMX	1--- 0--1
300006h	CONFIG4L	$\overline{\text{BKBUG}}$	XINST	BBSIZ1	BBSIZ0	—	—	—	STVREN	1000 ---1
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0	---- --11
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0	---- --11
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0	---- --11
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1[(1)]	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	见 表 19-2
3FFFFFh	DEVID2[(1)]	DEV10	DEV9	DEV8	DEV7	DEEV6	DEV5	DEV4	DEV3	见 表 19-2

图注： — = 未用 （读为 `0`）。阴影单元未用，读为 `0`。

注 1： DEVID 寄存器为只读寄存器，用户不能对其进行编程。

寄存器 19-1： **CONFIG1H：配置寄存器 1 的高字节 （字节地址为 300001h）**

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 **IESO**：内部 / 外部振荡器切换位
- 1 = 使能振荡器切换模式
- 0 = 禁止振荡器切换模式

bit 6 **FCMEN**：故障保护时钟监视器使能位
- 1 = 使能故障保护时钟监视器
- 0 = 禁止故障保护时钟监视器

bit 5-4 未用：读为 0

bit 3-0 **FOSC3:FOSC0**：振荡器选择位
- 11xx = 外部 RC 振荡器， RA6 用作 CLKO 引脚
- 101x = 外部 RC 振荡器， RA6 用作 CLKO 引脚
- 1001 = 内部振荡器电路， RA6 用作 CLKO 引脚， RA7 用作端口引脚
- 1000 = 内部振荡器电路， RA6、 RA7 均用作端口引脚
- 0111 = 外部 RC 振荡器， RA6 用作端口引脚
- 0110 = PLL 使能的 HS 振荡器 （时钟频率 = 4 × FOSC1）
- 0101 = EC 振荡器， RA6 用作端口引脚
- 0100 = EC 振荡器， RA6 用作 CLKO 引脚
- 0011 = 外部 RC 振荡器， RA6 用作 CLKO 引脚
- 0010 = HS 振荡器
- 0001 = XT 振荡器
- 0000 = LP 振荡器

寄存器 19-2： CONFIG2L：配置寄存器 2 的低字节 （字节地址为 300002h）

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	BORV1[1]	BORV0[1]	BOREN1[2]	BOREN0[2]	$\overline{\text{PWRTEN}}$[2]
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-5　未用：读为 0

bit 4-3　**BORV1:BORV0**：欠压复位门限电压位 [1]
　11 = 最大设置
　•
　•
　•
　00 = 最小设置

bit 2-1　**BOREN1:BOREN0**：欠压复位使能位 [2]
　11 = 仅由硬件使能欠压复位 （禁止 SBOREN）
　10 = 仅由硬件使能欠压复位，休眠模式下被禁止 （禁止 SBOREN）
　01 = 由软件使能和控制欠压复位 （使能 SBOREN）
　00 = 禁止使用硬件或软件使能欠压复位

bit 0　**$\overline{\text{PWRTEN}}$**：上电延时定时器使能位 [2]
　1 = 禁止 PWRT
　0 = 使能 PWRT

注　**1：** 请参见**第 22.1 节 “直流规范”**了解具体规范。
　2： 上电延时定时器与欠压复位是相互独立的，这样可以分别控制两者的操作。

寄存器 19-3： CONFIG2H：配置寄存器 2 的高字节 （字节地址为 300003h）

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-5 未用：读为 0

bit 4-1 **WDTPS3:WDTPS0**：看门狗定时器后分频比选择位
1111 = 1:32,768
1110 = 1:16,384
1101 = 1:8,192
1100 = 1:4,096
1011 = 1:2,048
1010 = 1:1,024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

bit 0 **WDTEN**：看门狗定时器使能位
1 = 使能 WDT
0 = 禁止 WDT （由 SWDTEN 位控制）

寄存器 19-4： CONFIG3H：配置寄存器 3 的低字节 （字节地址为 300005h）

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	U-0
—	—	—	—	HPOL[(1)]	LPOL[(1)]	PWMPIN	—
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-4 未用：读为 0

bit 3 **HPOL**：上端晶体管极性位 （奇数 PWM 输出极性控制位） [(1)]
1 = PWM1、 PWM3 和 PWM5 均为高电平有效 （默认状态）
0 = PWM1、 PWM3 和 PWM5 均为低电平有效

bit 2 **LPOL**：下端晶体管极性位 （偶数 PWM 输出极性控制位） [(1)]
1 = PWM0、 PWM2 和 PWM4 均为高电平有效 （默认状态）
0 = PWM0、 PWM2 和 PWM4 均为低电平有效

bit 2 **PWMPIN**：PWM 输出引脚复位状态控制位
1 = PWM 输出在复位时被禁止
0 = PWM 输出在复位时驱动有效的电平 [(2)]

bit 0 未用：读为 0

注 1： 极性控制位 HPOL 和 LPOL 定义 PWM 输出信号的有效和无效状态，以及因故障输入或手动更改 PWM 导致的 PWM 的状态。

2： 当 PWMPIN = 0 时， PWMEN<2:0> = 100。 PWM 输出极性由 HPOL 和 LPOL 定义。

寄存器 19-5： CONFIG3H：配置寄存器 3 的高字节 （字节地址为 300005h）

R/P-1	U-0	U-0	U-0	R/P-0	U-0	U-0	R/P-1
MCLRE	—	—	—	T1OSCMX	—	—	FLTAMX
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 **MCLRE**：$\overline{\text{MCLR}}$ 引脚使能位
- 1 = 使能 $\overline{\text{MCLR}}$ 引脚，禁止 RA5 输入引脚
- 0 = 使能 RA5 输入引脚，禁止 $\overline{\text{MCLR}}$ 引脚

bit 6-4 未用：读为 0

bit 3 **T1OSCMX**：T1OSO/T1CKI 复用位
- 1 = T1OSO/T1CKI 引脚与 RA6 复用
- 0 = T1OSO/T1CKI 引脚与 RB2 复用

bit 2-1 未用：读为 0

bit 0 **FLTAMX**：$\overline{\text{FLTA}}$ 复用位
- 1 = $\overline{\text{FLTA}}$ 与 RA5 复用
- 0 = $\overline{\text{FLTA}}$ 与 RA7 复用

寄存器 19-6： CONFIG4L：配置寄存器 4 的低字节 （字节地址为 300006h）

R/P-1	R/P-0	R/P-0	R/P-0	U-0	U-0	U-0	R/P-1
$\overline{\text{BKBUG}}$	XINST	BBSIZ1	BBSIZ0	—	—	—	STVREN
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 **DEBUG**：后台调试器使能位
1 = 禁止后台调试器， RB6 和 RB7 被配置为通用 I/O 引脚
0 = 使能后台调试器， RB6 和 RB7 专用于在线调试

bit 6 **XINST**：扩展指令集使能位
1 = 使能指令集扩展和变址寻址模式
0 = 禁止指令集扩展和变址寻址模式

bit 5-4 **BBSIZ<1:0>**：引导区大小选择位
对于 PIC18F1330 器件：
11 = 1 kW 引导区
10 = 1 kW 引导区
01 = 512W 引导区
00 = 256W 引导区
对于 PIC18F1230 器件：
11 = 512W 引导区
10 = 512W 引导区
01 = 512W 引导区
00 = 256W 引导区

bit 3-1 未用：读为 0

bit 0 **STVREN**：堆栈上溢 / 下溢复位使能位
1 = 使能堆栈上溢 / 下溢复位
0 = 禁止堆栈上溢 / 下溢复位

寄存器 19-7： CONFIG5L：配置寄存器 5 的低字节 （字节地址为 300008h）

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	CP1	CP0
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-2 未用：读为 0

bit 1 **CP1**：代码保护位 （Block 1 程序存储区）
1 = Block 1 无代码保护
0 = Block 1 有代码保护

bit 0 **CP0**：代码保护位 （Block 0 程序存储区）
1 = Block 0 无代码保护
0 = Block 0 有代码保护

寄存器 19-8： CONFIG5H：配置寄存器 5 的高字节 （字节地址为 300009h）

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 **CPD**：数据 EEPROM 代码保护位
1 = 数据 EEPROM 无代码保护
0 = 数据 EEPROM 有代码保护

bit 6 **CPB**：引导存储区代码保护位
1 = 引导区无代码保护
0 = 引导区有代码保护

bit 5-0 未用：读为 0

 超前信息

寄存器 19-9： CONFIG6L：配置寄存器 6 的低字节 （字节地址为 30000Ah）

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	WRT1	WRT0
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-2 未用：读为 0

bit 1 **WRT1**：写保护位 （Block 1 程序存储区）
1 = Block 1 无写保护
0 = Block 1 有写保护

bit 0 **WRT0**：写保护位 （Block 0 程序存储区）
1 = Block 0 无写保护
0 = Block 0 有写保护

寄存器 19-10： CONFIG6H：配置寄存器 6 的高字节 （字节地址为 30000Bh）

R/C-1	R/C-1	R-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC[(1)]	—	—	—	—	—
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 **WRTD**：数据 EEPROM 写保护位
1 = 数据 EEPROM 无写保护
0 = 数据 EEPROM 有写保护

bit 6 **WRTB**：引导存储区写保护位
1 = 引导区无写保护
0 = 引导区有写保护

bit 5 **WRTC**：配置寄存器写保护位 [(1)]
1 = 配置寄存器无写保护
0 = 配置寄存器有写保护

bit 4-0 未用：读为 0

注 1： 在正常执行模式下，该位只读；仅在编程模式下该位才可写入。

寄存器 19-11： CONFIG7L：配置寄存器 7 的低字节 （字节地址为 30000Ch）

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	EBTR1	EBTR0
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-2 未用：读为 0

bit 1 **EBTR1：**表读保护位 （Block 1 程序存储区）
1 = Block 1 无表读保护，可从其他区块对其执行表读操作
0 = Block 1 有表读保护，不能从其他区块对其执行表读操作

bit 0 **EBTR0：**表读保护位 （Block 0 程序存储区）
1 = Block 0 无表读保护，可从其他区块对其执行表读操作
0 = Block 0 有表读保护，不能从其他区块对其执行表读操作

寄存器 19-12： CONFIG7H：配置寄存器 7 的高字节 （字节地址为 30000Dh）

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7							bit 0

图注：		
R = 可读位	C = 可清零位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7 未用：读为 0

bit 6 **EBTRB：**引导存储区表读保护位
1 = 引导区无表读保护，可从其他区块对其执行表读操作
0 = 引导区有表读保护，不能从其他区块对其执行表读操作

bit 5-0 未用：读为 0

寄存器 19-13： DEVID1：PIC18F1230/1330 器件的 ID 寄存器 1

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

图注：		
R = 只读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-5 **DEV2:DEV0**：器件 ID 位
这些位与 DEVID2 寄存器中的 DEV10:DEV3 位一起用于标识器件编号。

bit 4-0 **REV3:REV0**：版本 ID 位
这些位用于表明器件的版本。

寄存器 19-14： DEVID2：PIC18F1230/1330 器件的 ID 寄存器 2

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

图注：		
R = 只读位	P = 可编程位	U = 未用位，读为 0
-n = 未对器件编程时的值		u = 编程后状态不变

bit 7-0 **DEV10:DEV3**：器件 ID 位
这些位与 DEVID1 寄存器中的 DEV2:DEV0 一起用于标识器件编号。

19.2 看门狗定时器 （WDT）

PIC18F1230/1330 器件的 WDT 由 INTRC 时钟源驱动。当使能 WDT 时，也将同时使能时钟源。WDT 超时溢出周期的标称值为 4 ms，其稳定性与 INTRC 振荡器相同。

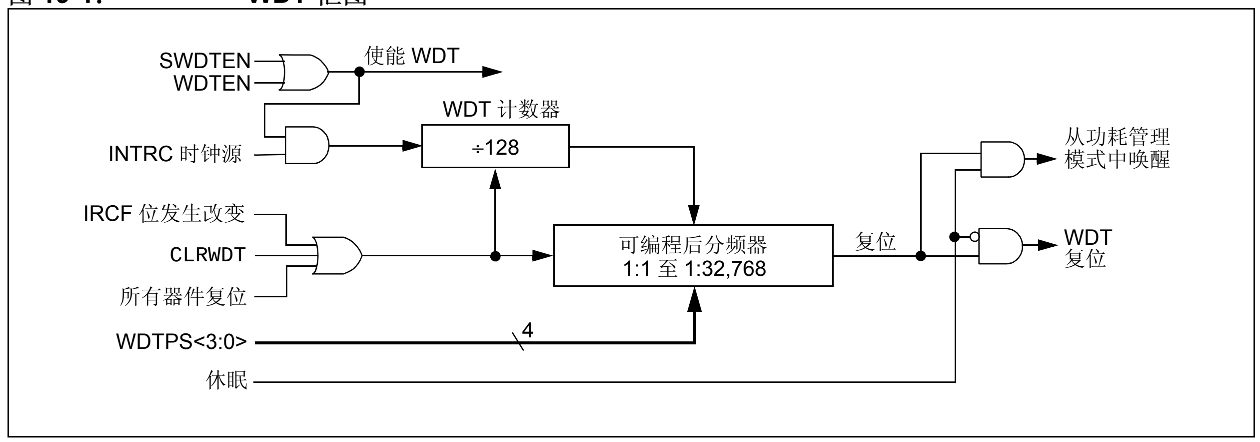
4 ms 的 WDT 周期与 16 位后分频器中的值相乘。通过配置寄存器 2H 中的位控制一个多路开关可以对 WDT 后分频器的输出进行选择。因此可获得范围为 4 ms 至 131.072 秒（2.18 分钟）的超时周期。当发生以下任一事件时， WDT 和后分频器将被清零，这些事件包括 ：执行 SLEEP 或 CLRWDT 指令、IRCF 位（OSCCON<6:4>）发生改变或发生时钟故障。

注	
1：	CLRWDT 和 SLEEP 指令会清零 WDT 和后分频器。
2：	更改 IRCF 位 （OSCCON<6:4>）的设置会清零 WDT 和后分频器的计数值。
3：	当执行 CLRWDT 指令时，后分频器的计数值将被清零。

19.2.1 控制寄存器

寄存器 19-15 所示为 WDTCON 寄存器。这是一个可读写的寄存器，它包含一个控制位，仅当禁止 WDT 时，应用软件才能使用该控制位来控制 WDT。

图 19-1： WDT 框图



 超前信息

寄存器 19-15： WDTCON：看门狗定时器控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN[(1)]
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = 上电复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 7-1 未用：读为 0

bit 0 **SWDTEN**：看门狗定时器软件控制使能位 [(1)]
1 = 使能看门狗定时器
0 = 禁止看门狗定时器

注 1： 当使能 WDTEN 配置位时，该位不起作用。

表 19-2： 看门狗定时器的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在的页
RCON	IPEN	SBOREN[(1)]	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	42
WDTCON	—	—	—	—	—	—	—	SWDTEN[(2)]	42

图注： — = 未用 （读为 0）。看门狗定时器不使用阴影单元。

注 1： 只有当 BOREN1:BOREN0 配置位 = 01 时， SBOREN 位才可用，否则该位被禁止且读为 0。请参见 **第 4.4 节 “欠压复位 （BOR）”**。

2： 当使能 WDTEN 配置位时，该位不起作用。

19.3 双速启动

双速启动功能允许单片机在主时钟源可用之前使用 INTOSC 振荡器作为时钟源，从而帮助器件最大限度地缩短从振荡器起振到代码执行之间的延时。通过将 IESO 配置位置 1 可使能该功能。

仅当主振荡器模式为 LP、XT、HS 或 HSPLL （基于晶振的模式）时才可使用双速启动。其他时钟源不需要 OST 起振延时。对于这些时钟源，应禁止双速启动。

一旦使能双速启动，在发生上电复位（如果已使能）或从休眠模式唤醒时，在上电延时定时器延时结束之后，器件将使用内部振荡器作为时钟源。这样几乎可使代码在主振荡器起振、OST 运行的同时立即执行。一旦 OST 超时，器件就自动切换到 PRI_RUN 模式。

为了在唤醒器件时使用更快的时钟速率，可以选择 INTOSC 或后分频器时钟源，这可以通过在复位发生后立即设置 IRCF2:IRCF0 实现。对于从休眠模式唤醒的情况，可以通过在进入休眠模式之前设置 IRCF2:IRCF0 来选择 INTOSC 或后分频器时钟源。

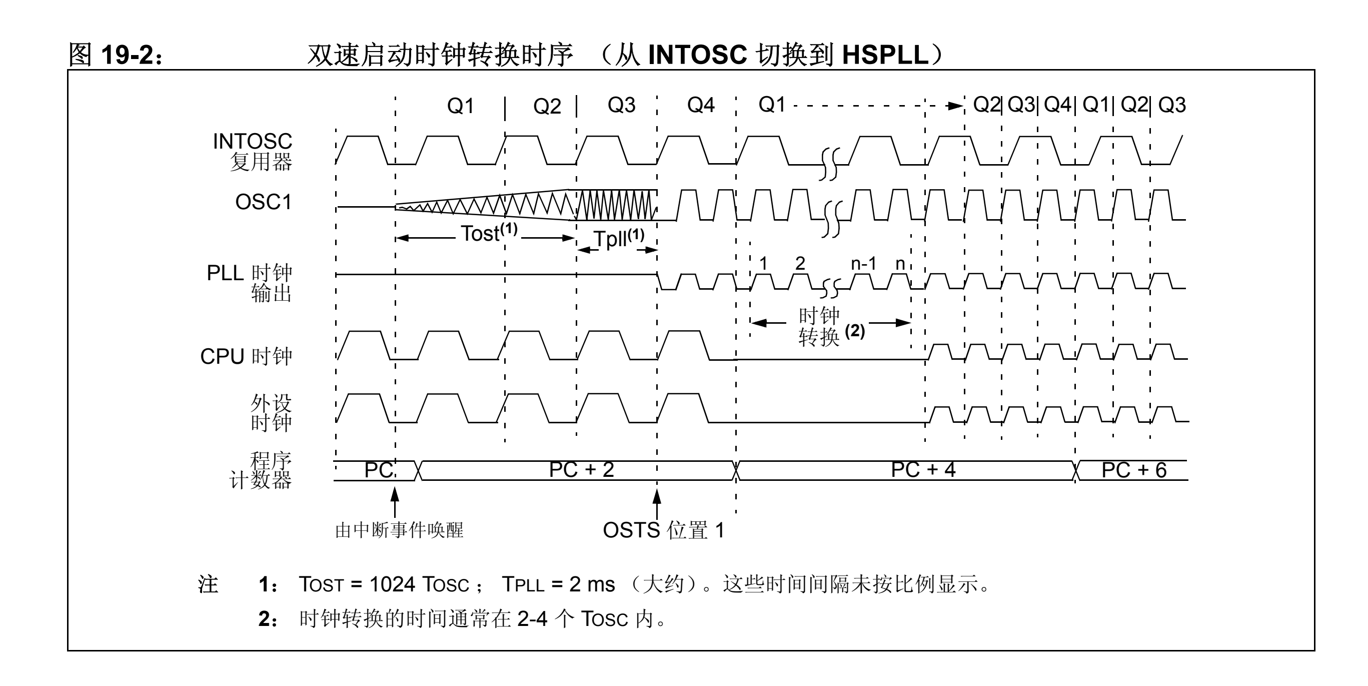
在其他的功耗管理模式下不使用双速启动。器件将使用当前选定的时钟源直到主时钟源可用为止。该操作与 IESO 位的设置无关。

19.3.1 使用双速启动时的注意事项

当在双速启动中使用 INTOSC 振荡器时，器件仍将遵守进入功耗管理模式（包括执行多条 SLEEP 指令）的正常指令顺序 （见**第 3.1.4 节 “多条 Sleep 命令”**）。实际上，这意味着在 OST 超时前用户代码可以改变 SCS1:SCS0 位的设置或执行 SLEEP 指令。这就使应用程序能短暂地唤醒器件，执行 “日常事务”子程序，并在器件开始使用主时钟源前返回休眠状态。

用户代码还能通过查询 OSTS 位 （OSCCON<3>）的状态来确定当前主时钟源是否正在为系统提供时钟。若该位置 1，则表示主振荡器正在为系统提供时钟。否则，表示当器件从复位或休眠模式唤醒期间由内部振荡器电路为系统提供时钟。

图 19-2： 双速启动时钟转换时序 （从 **INTOSC** 切换到 **HSPLL**）



注 1：TOST = 1024 TOSC ； TPLL = 2 ms （大约）。这些时间间隔未按比例显示。
2：时钟转换的时间通常在 2-4 个 TOSC 内。

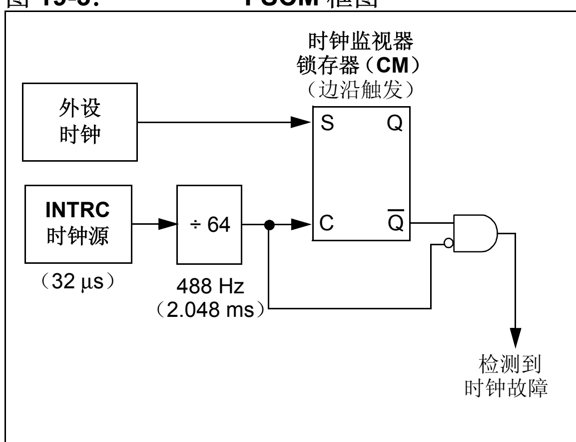
 超前信息

19.4 故障保护时钟监视器

故障保护时钟监视器（FSCM）可使单片机在发生外部时钟故障时，自动将系统时钟切换到内部振荡器电路以维持器件继续运行。将FCMEN配置位置1可使能FSCM功能。

当使能 FSCM 时， INTRC 振荡器将一直保持运行以监视外设时钟，并且在外设时钟发生故障时立即提供备用时钟。时钟监视（如图 19-3 所示）通过创建一个采样时钟信号实现，该信号为 INTRC 输出的 64 分频信号。这样就使得FSCM采样时钟相邻脉冲之间有充足的时间间隔，从而保证在此间隔期间必然会出现外设时钟边沿。外设时钟和采样时钟作为时钟监视器锁存器（CM）的输入。 CM 在外设时钟的下降沿被置 1，在采样时钟的上升沿被清零。

图 19-3： FSCM 框图



在采样时钟的下降沿检测时钟故障。如果在出现采样时钟的下降沿时， CM 仍置 1，就表示检测到外部时钟故障（图 19-4）。这将引发以下事件：

- FSCM 将 OSCFIF （PIR2<7>）置 1，产生振荡器故障中断。
- 器件时钟源切换为内部振荡器电路（OSCCON 不会被更新，因此无法显示当前时钟源——这就是故障保护状态）。
- WDT 复位。

切换过程中，对于时序要求较高的应用，内部振荡器电路的后分频频率可能不够稳定。在这些情况下，最好选择另一种时钟配置并进入其他功耗管理模式。可以尝试部分恢复或执行受控的关闭。请参见**第 3.1.4 节“多条 Sleep 命令”**和**第 19.3.1 节“使用双速启动时的注意事项”**了解更多详细信息。

为了在唤醒器件时使用更快的时钟速率，可以选择 INTOSC 或后分频器时钟源，这可以通过在复位发生后立即设置 IRCF2:IRCF0 实现。对于从休眠模式唤醒的情况，可以在进入休眠模式之前设置 IRCF2:IRCF0 来选择 INTOSC 或后分频器时钟源。

FSCM 只能检测出主时钟源或辅助时钟源的故障。如果内部振荡器电路发生故障，则无法检测到，当然也不会采取任何措施。

19.4.1 FSCM 和看门狗定时器

FSCM 和 WDT 均以 INTRC 振荡器作为时钟源。由于 WDT 使用独立的分频器和计数器，当使能 FSCM 时，禁止 WDT 对 INTRC 振荡器的运行没有影响。

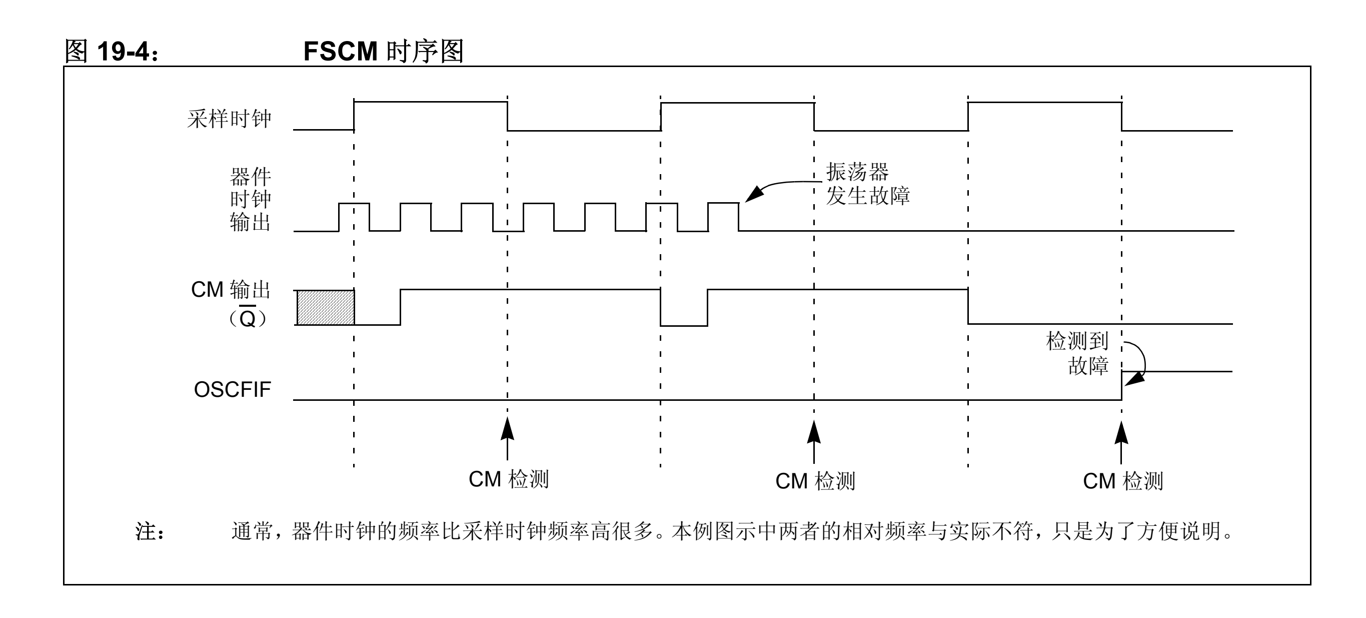
如前所述，当发生时钟故障时，时钟源将切换到 INTOSC 时钟源。根据 IRCF2:IRCF0 位选择的频率的不同，代码执行速度也会相应发生很大的变化。如果用小的预分频值使能 WDT，时钟速率的下降将导致 WDT 发生超时并在随后使器件复位。由于这个原因，故障保护事件也会使 WDT 和后分频器复位，使 WDT 从运行速率发生变化的时刻开始重新计数，从而减少发生错误超时的可能性。

19.4.2 退出故障保护运行模式

器件复位或进入功耗管理模式均可终止故障保护状态。发生复位时，控制器启动在配置寄存器 1H 中指定的主时钟源（伴有如 OST 或 PLL 定时器等振荡器模式所需的起振延时）。 INTOSC 复用器将提供系统时钟直到主时钟源就绪为止（类似于双速启动）。当主时钟源可用时，系统时钟源将切换回主时钟（OSCCON 寄存器中的 OSTS 位置 1，表明当前使用的是主时钟源）。然后，故障保护时钟监视器恢复对外设时钟的监视。

在起振期间，主时钟源可能永远不能就绪。在这种情况下，器件将以 INTOSC 复用器作为时钟源。 OSCCON 寄存器将保持复位状态直到进入功耗管理模式为止。

图 19-4： FSCM 时序图



注： 通常，器件时钟的频率比采样时钟频率高很多。本例图示中两者的相对频率与实际不符，只是为了方便说明。

19.4.3 功耗管理模式下的 FSCM 中断

进入功耗管理模式时，时钟多路开关选择由 OSCCON 寄存器选定的时钟源。在功耗管理模式下将恢复对功耗管理时钟源的故障保护监视。

如果在功耗管理模式下发生了振荡器故障，随后的操作将取决于是否允许振荡器故障中断。如果允许（OSCFIF = 1），代码执行将以 INTOSC 复用器作为时钟源，并且不会自动转回到发生故障的时钟源。

如果禁止该中断，空闲模式下振荡器故障所导致的中断将使 CPU 开始执行指令，此时由 INTOSC 时钟源为 CPU 提供时钟。

19.4.4 上电复位或从休眠中唤醒

FSCM 用于在器件退出上电复位（POR）或低功耗休眠模式后来检测振荡器故障。当系统主时钟为 EC、RC 或 INTRC 模式时，对时钟源的监视会在这些事件发生后立即开始。

对于涉及到晶振或谐振器的振荡器模式（如 HS、HSPLL、LP 或 XT），情况会有些不同。由于这类振荡器需要的起振时间可能比 FCSM 采样时钟周期长很多，因此可能会检测到假时钟故障。为了避免这一情况，内部振荡器电路会被自动配置为器件时钟直到主时钟稳定下来为止（OST 和 PLL 定时器已完成延时）。这与双速启动模式相同。一旦主时钟稳定下来，INTRC 就将重新作为 FSCM 时钟源。

> **注：** 用于防止上电复位或从休眠状态唤醒时发生假中断的逻辑电路，同样也将阻止在发生这些事件后对振荡器故障的检测。通过监视 OSTS 位，并使用定时程序来确定振荡器起振时间是否过长可避免这个问题。即便如此，在检测到振荡器故障时也不会将中断标志位置 1。

正如**第 19.3.1 节“使用双速启动时的注意事项”**中所述，在等待主时钟稳定的过程中，可以选择另一种时钟配置并进入某一功耗管理模式。当选择新的功耗管理模式时，主时钟将被禁止。

19.5 程序校验和代码保护

PIC18 闪存器件的整个代码保护结构与其他的 PIC® 系列器件有很大的区别。

用户程序存储器被分为三个存储区。其中一个存储区为大小可变的引导区（最大为 2KB），剩余部分被分为两个按二进制边界划分的存储区。

每个存储区都有三个代码保护位与之相关联。它们是：

- 代码保护位（CPx）
- 写保护位（WRTx）
- 外部存储区表读位（EBTRx）

图 19-5 显示了 4 KB 和 8 KB 器件的程序存储器构成，以及与每个存储区相关的特定代码保护位。表 19-3 总结了这些位的具体地址。

图 19-5：　PIC18F1230/1330 的代码保护程序存储器

存储容量 / 器件

4 KB（PIC18F1230）	8 KB（PIC18F1330）	地址范围	存储区代码保护控制位
引导区	引导区	000000h 0003FFh	CPB、 WRTB 和 EBTRB
Block 0	引导区	000400h 0007FFh	CP0、 WRT0 和 EBTR0
Block 1	Block 0	000800h 000FFFh	CP1、 WRT1 和 EBTR1
未用 读为 0	Block 1	001000h 001FFFh	CP2、 WRT2 和 EBTR2
未用 读为 0	未用 读为 0	002000h 1FFFFFh	（未用的存储空间）

表 19-3：　代码保护寄存器汇总

寄存器名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

图注：　未使用阴影单元。

19.5.1 程序存储器代码保护

使用表读和表写指令可以对程序存储器的任何单元进行读写操作。器件 ID 可由表读指令读取。也可使用表读和表写指令对配置寄存器进行读写操作。

在正常执行模式下，CPx 位不起任何作用，它用于禁止来自外部的读写操作。如果 WRTx 配置位是 0，即可保护用户存储区不受表写指令的影响。而 EBTRx 位控制表读操作。如果用户存储区中某一区块的 EBTRx 位为 0，允许在该存储区内部使用表读指令执行读操作，但不允许其他存储区对该存储区执行表读操作，否则读出的结果为 0。图 19-6 到图 19-8 说明了表读和表写保护的操作。

注：	代码保护位仅能从 1 状态改写为 0 状态，而不可能从 0 状态改写到 1 状态。只有使用整个芯片擦除或块擦除功能才能将代码保护位置 1。而上述功能又仅能通过 ICSP 操作或外部编程器启用。

图 19-6： 禁止表写操作 （WRTx）

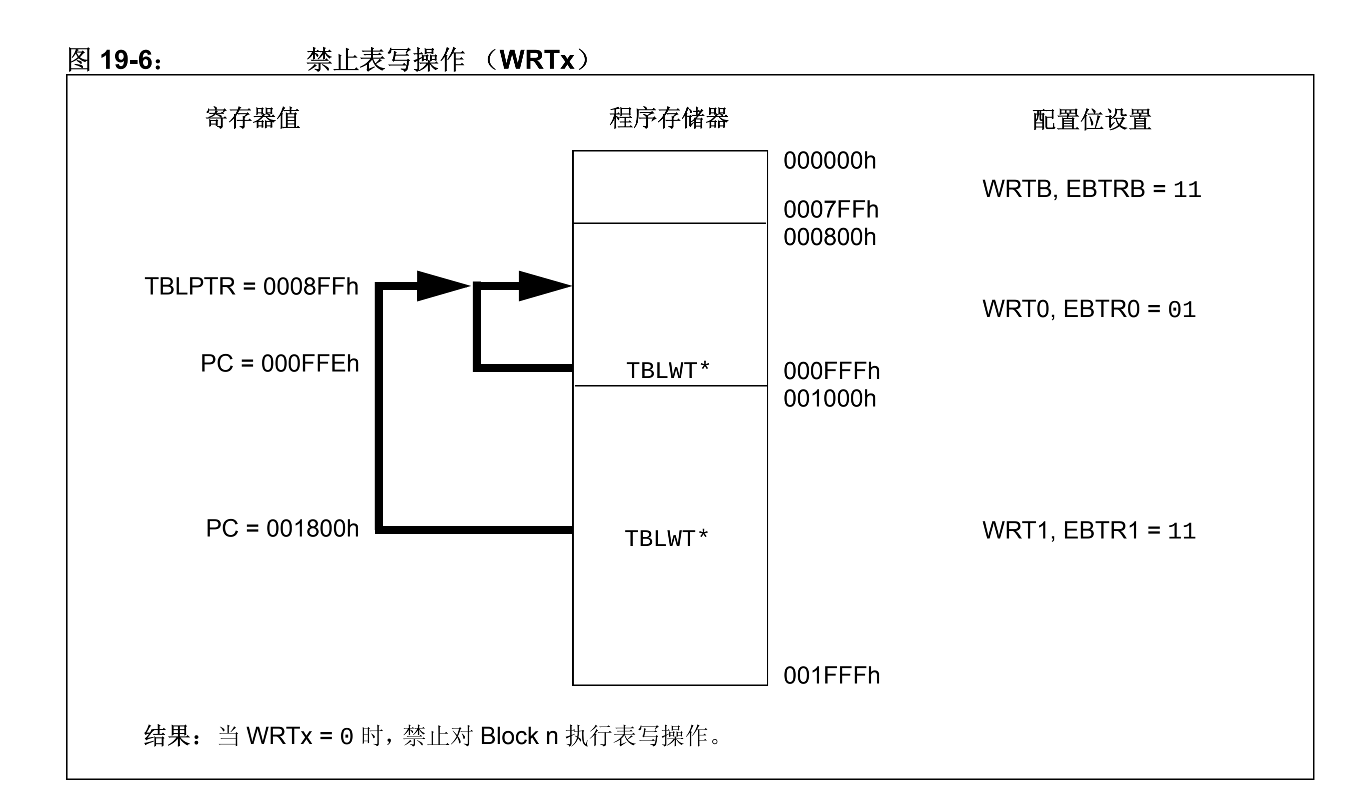


图 **19-7**： 禁止外部存储区表读操作 （**EBTRx**）

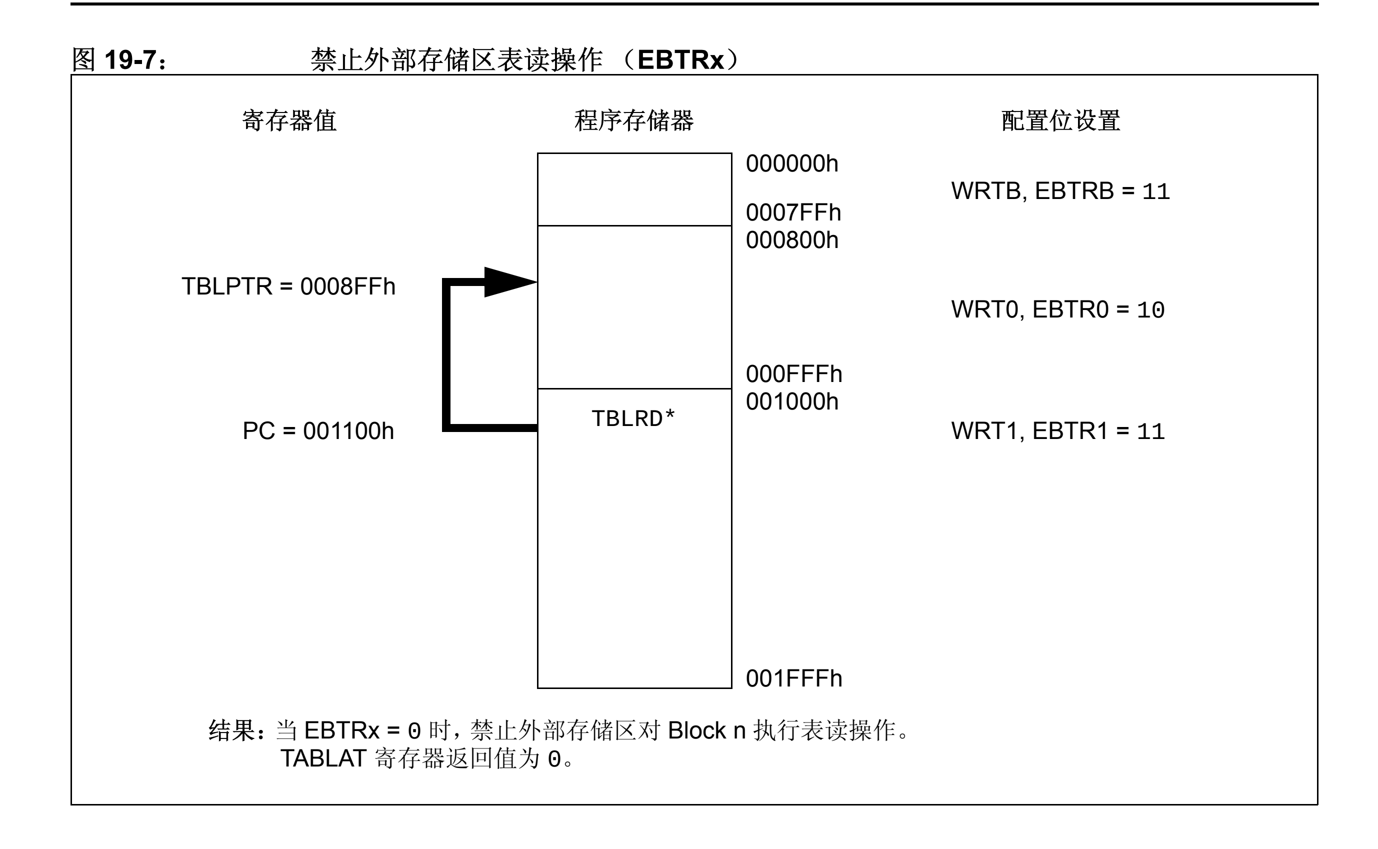
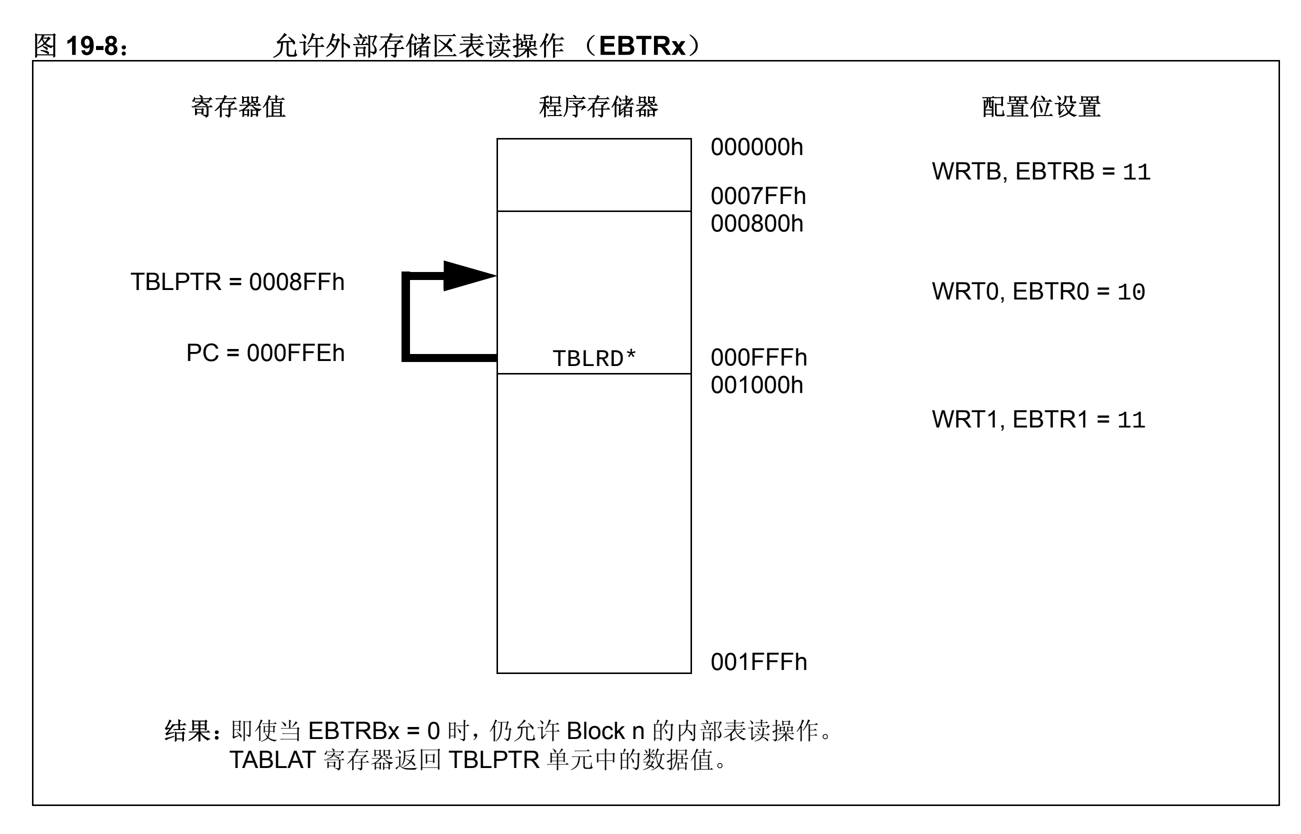


图 **19-8**： 允许外部存储区表读操作 （**EBTRx**）



19.5.2 数据 EEPROM 代码保护

整个数据 EEPROM 的外部读写保护由 CPD 和 WRTD 这两个位控制。CPD 禁止外部读写数据 EEPROM。WRTD 禁止内部和外部写入数据 EEPROM。不管保护位如何设置，CPU 在正常运行过程中始终能够读数据 EEPROM。

19.5.3 配置寄存器保护

配置寄存器可以被写保护。WRTC 位控制对配置寄存器的写保护。在正常执行模式下，WRTC 位是只读的。WRTC 位仅能通过 ICSP 操作或外部编程器写入。

19.6 ID 单元

有 8 个存储单元（200000h-200007h）被指定为 ID 单元，供用户存储校验和或其他代码标识。在正常执行期间可通过 TBLRD 和 TBLWT 指令读写这些单元；也可通过编程 / 验证读写这些单元。当器件有代码保护时，仍可读取 ID 单元。

19.7 在线串行编程

PIC18F1230/1330 系列单片机可以在最终应用电路中进行串行编程。只需要 5 根线即可完成这一操作，其中时钟线、数据线各一根，其余 3 根分别是电源线、地线和编程电压线。这允许用户使用未编程器件制造电路板，仅在产品交付前才对单片机进行编程。从而允许将最新版本的固件或定制固件烧写入器件。

19.8 在线调试器

将 $\overline{\text{DEBUG}}$ 配置位编程为 0 可使能在线调试功能。这一功能允许使用 MPLAB® IDE 进行一些简单的调试。当使能了单片机的这项功能时，某些资源就不再是通用的了。表 19-4 显示了后台调试器所需的资源。

表 19-4： 调试器资源

I/O 引脚：	RB6 和 RB7
堆栈：	2 级
程序存储器：	512 个字节
数据存储器：	10 个字节

要使用单片机的在线调试器功能，用户在设计中就必须实现与 $\overline{\text{MCLR}}$/VPP/RA5/$\overline{\text{FLTA}}$、VDD、VSS、RB7/PWM5/PGD 和 RB6/PWM4/PGC 的在线串行编程连接，从而可与 Microchip 或第三方开发工具公司提供的在线调试器模块交互。

19.9 单电源 ICSP 编程

PIC18F1230/1330 系列器件不支持低电压 ICSP 编程或 LVP。此系列只能采用高电压 ICSP 编程。更多详细信息请参见*"PIC18F1230/1330 Flash Microcontroller Programming Specification"*（DS39752）。

没有代码保护的存储器可用块擦除或逐行擦除的方法来擦除，然后可在指定的 VDD 电压下对其进行写操作。如果要擦除有代码保护的存储器，则必须使用块擦除的方法。

20.0 开发支持

一系列硬件及软件开发工具对 PIC® 单片机提供支持：

- 集成开发环境
  - MPLAB® IDE 软件
- 汇编器 / 编译器 / 链接器
  - MPASM™ 汇编器
  - MPLAB C18 和 MPLAB C30 C 编译器
  - MPLINK™ 目标链接器 / MPLIB™ 目标库管理器
  - MPLAB ASM30 汇编器 / 链接器 / 库
- 模拟器
  - MPLAB SIM 软件模拟器
- 仿真器
  - MPLAB ICE 2000 在线仿真器
  - MPLAB REAL ICE™ 在线仿真器
- 在线调试器
  - MPLAB ICD 2
- 器件编程器
  - PICSTART® Plus 开发编程器
  - MPLAB PM3 器件编程器
  - PICkit™ 2 开发编程器
- 低成本演示和开发板及评估工具包

20.1 MPLAB 集成开发环境软件

MPLAB IDE 软件为 8/16 位单片机市场提供了前所未有的易于使用的软件开发平台。MPLAB IDE 是基于 Windows® 操作系统的应用软件，包括：

- 一个包含所有调试工具的图形界面
  - 模拟器
  - 编程器 （单独销售）
  - 仿真器 （单独销售）
  - 在线调试器 （单独销售）
- 具有彩色上下文代码显示的全功能编辑器
- 多项目管理器
- 内容可直接编辑的可定制式数据窗口
- 高级源代码调试
- 可视化器件初始化程序，便于进行寄存器的初始化
- 鼠标停留在变量上进行查看的功能
- 通过拖放把变量从源代码窗口拉到观察窗口
- 丰富的在线帮助
- 集成了可选的第三方工具，如 HI-TECH 软件 C 编译器和 IAR C 编译器

MPLAB IDE 可以让您：

- 编辑源文件 （汇编语言或 C 语言）
- 点击一次即可完成汇编 （或编译）并将代码下载到 PIC MCU 仿真器和模拟器工具中 （自动更新所有项目信息）
- 可使用如下各项进行调试：
  - 源文件 （汇编语言或 C 语言）
  - 混合汇编语言和 C 语言
  - 机器码

MPLAB IDE 在单个开发范例中支持使用多种调试工具，包括从成本效益高的模拟器到低成本的在线调试器，再到全功能的仿真器。这样缩短了用户升级到更加灵活而功能更强大的工具时的学习时间。

20.2 MPASM 汇编器

MPASM 汇编器是全功能通用宏汇编器，适用于所有的 PIC MCU。

MPASM 汇编器可生成用于 MPLINK 目标链接器的可重定位目标文件、Intel® 标准 HEX 文件、详细描述存储器使用状况和符号参考的 MAP 文件、包含源代码行及生成机器码的绝对 LST 文件以及用于调试的 COFF 文件。

MPASM 汇编器具有如下特征：

- 集成在 MPLAB IDE 项目中
- 用户定义的宏可简化汇编代码
- 对多用途源文件进行条件汇编
- 允许完全控制汇编过程的指令

20.3 MPLAB C18 和 MPLAB C30 C 编译器

MPLAB C18 和 MPLAB C30 代码开发系统是完全的 ANSI C 编译器，分别适用于 Microchip 的 PIC18 系列单片机及 dsPIC30F、dsPIC33 和 PIC24 系列数字信号控制器。这些编译器可提供其他编译器并不具备的强大的集成功能和出众的代码优化能力，且使用方便。

为便于源代码调试，编译器提供了针对 MPLAB IDE 调试器的优化符号信息。

20.4 MPLINK 目标链接器 / MPLIB 目标库管理器

MPLINK 目标链接器包含了由 MPASM 汇编器、MPLAB C18 C 编译器产生的可重定位目标。通过使用链接器脚本中的指令，它还可链接预编译库中的可重定位目标。

MPLIB目标库管理器管理预编译代码库文件的创建和修改。当从源文件调用库中的一段子程序时，只有包含此子程序的模块被链接到应用中。这样可使大型库在许多不同应用中被高效地利用。

目标链接器 / 库管理器具有如下特征：

- 高效地连接单个的库而不是许多小文件
- 通过将相关的模块组合在一起来增强代码的可维护性
- 只要列出、替换、删除和抽取模块，便可灵活地创建库

20.5 MPLAB ASM30 汇编器、链接器和库管理器

MPLAB ASM30 汇编器为 dsPIC30F 器件提供转换自符号汇编语言的可重定位机器码。 MPLAB C30 C 编译器使用该汇编器生成目标文件。汇编器产生可重定位目标文件之后，可将这些目标文件存档，或与其他可重定位目标文件和存档链接以生成可执行文件。该汇编器有如下显著特征：

- 支持整个 dsPIC30F 指令集
- 支持定点数据和浮点数据
- 命令行界面
- 丰富的指令集
- 灵活的宏语言
- MPLAB IDE 兼容性

20.6 MPLAB SIM 软件模拟器

MPLAB SIM软件模拟器在指令级对PIC MCU和dsPIC® DSC 进行模拟，使得用户可以在 PC 主机的环境下进行代码开发。对于任何给定的指令，用户均可对数据区进行检查或修改，并通过各种触发机制来产生激励。可以将各寄存器的情况记录在文件中，以便进行进一步地运行时分析。跟踪缓冲器和逻辑分析器的显示使模拟器还能记录和跟踪程序的执行、 I/O 的动作、大部分的外设及内部寄存器的状况。

MPLAB SIM 软件模拟器完全支持使用 MPLAB C18 和 MPLAB C30 C 编译器以及MPASM和MAPLAB ASM30 汇编器的符号调试。该软件模拟器可用于在硬件实验室环境外灵活地开发和调试代码，是一款完美且经济的软件开发工具。

20.7 MPLAB ICE 2000 高性能在线仿真器

MPLAB ICE 2000 在线仿真器旨在为产品开发工程师提供一整套用于 PIC 单片机的设计工具。 MPLAB ICE 2000 在线仿真器的软件控制由 MPLAB 集成开发环境平台提供，它允许在单一环境下进行编辑、编译、下载以及源代码调试。

MPLAB ICE 2000 是全功能仿真器系统，它具有增强的跟踪、触发和数据监控功能。处理器模块可插拔，使系统可轻松进行重新配置以适应各种不同处理器的仿真需要。MPLAB ICE 2000 在线仿真器的架构允许对其进行扩展以支持新的 PIC 单片机。

MPLAB ICE 2000 在线仿真器系统设计为一款实时仿真系统，该仿真系统具备通常只有昂贵的开发工具中才有的高级功能。选择 PC 平台和 Microsoft® Windows® 32 位操作系统可使这些功能在一个简单而统一的应用中得到很好的利用。

20.8 MPLAB REAL ICE 在线仿真器系统

MPLAB REAL ICE在线仿真器系统是Microchip针对其闪存 DSC® 和 MCU 器件而推出的新一代高速仿真器。结合 MPLAB 集成开发环境（IDE）所具有的易于使用且功能强大的图形用户界面，该仿真器可对 PIC® 和 dsPIC® 闪存单片机进行调试和编程。IDE 是随每个工具包一起提供的。

MPLAB REAL ICE 探针通过高速 USB 2.0 接口与设计工程师的 PC 相连，并利用与常用 MPLAB ICD 2 系统兼容的连接器（RJ11）或新型抗噪声、高速低压差分信号（LVDS）互连电缆（CAT5）与目标板相连。

可通过 MPLAB IDE 下载将来版本的固件，对 MPLAB REAL ICE 进行现场升级。在即将推出的 MPLAB IDE 版本中，会支持许多新器件，还将增加一些新特性，如软件断点和汇编代码跟踪等。在同类仿真器中， MPLAB REAL ICE 的优势十分明显：低成本、高速仿真、实时变量监视、跟踪分析、复杂断点、耐用的探针接口及较长（长达 3 米）的互连电缆。

20.9 MPLAB ICD 2 在线调试器

Microchip 的在线调试器 MPLAB ICD 2 是一款功能强大而成本低廉的运行时开发工具，通过 RS-232 或高速 USB 接口与 PC 主机相连。该工具基于闪存 PIC MCU，可用于开发本系列及其他 PIC MCU 和 dsPIC DSC。MPLAB ICD 2 使用了闪存器件中内建的在线调试功能。该功能结合 Microchip 的在线串行编程（In-Circuit Serial Programming™， ICSP™）协议，可在 MPLAB 集成开发环境的图形用户界面上提供成本效益很高的在线闪存调试。这使设计人员可通过设置断点、单步运行以及对变量、CPU 状态以及外设寄存器进行监视的方法实现源代码的开发和调试。其全速运行特性可对硬件和应用进行实时测试。 MPLAB ICD 2 还可用作某些 PIC 器件的开发编程器。

20.10 MPLAB PM3 器件编程器

MPLAB PM3 器件编程器是一款通用的、符合 CE 规范的器件编程器，其可编程电压设置在 V_{DDMIN} 和 V_{DDMAX} 之间时可靠性最高。它有一个用来显示菜单和错误信息的大 LCD 显示器（128 x 64），以及一个支持各种封装类型的可拆卸模块化插槽装置。编程器标准配置中带有一根 ICSP™ 电缆。在单机模式下， MPLAB PM3 器件编程器不必与 PC 相连即可对 PIC 器件进行读取、验证和编程。在该模式下它还可设置代码保护。 MPLAB PM3 通过 RS-232 或 USB 电缆连接到 PC 主机上。MPLAB PM3 具备高速通信能力以及优化算法，可对存储器很大的器件进行快速编程，它还采用 SD/MMC 卡用作文件存储及数据安全应用。

20.11 PICSTART Plus 开发编程器

PICSTART Plus开发编程器是一款易于使用而成本低廉的原型编程器。它通过 COM（RS-232）端口与 PC 相连。 MPLAB 集成开发环境软件使得该编程器的使用简便、高效。PICSTART Plus 开发编程器支持采用 DIP 封装的大部分 PIC 器件，其引脚数最多可达 40 个。引脚数更多的器件，如 PIC16C92X 和 PIC17C76X，可通过连接一个转接插槽来获得支持。PICSTART Plus 开发编程器符合 CE 规范。

20.12 PICkit 2 开发编程器

PICkit™ 2 开发编程器是一个低成本编程器；对于某些选定闪存器件，它也是一个调试器，通过其易于使用的接口可对众多 Microchip 的低档、中档和 PIC18F 系列闪存单片机进行编程。 PICkit 2 入门工具包中包含一个有实验布线区的开发板、十二堂系列课程、软件和 HI-TECH 的 PICC™ Lite C 编译器，有助于用户快速掌握 PIC® 单片机的使用。这一工具包为使用 Microchip 功能强大的中档闪存系列单片机进行编程、评估和应用开发，提供了所需的一切。

20.13 演示、开发和评估板

有许多演示、开发和评估板可用于各种 PIC MCU 和 dsPIC DSC，实现对全功能系统的快速应用开发。大多数的演示、开发和评估板都有实验布线区，供用户添加定制电路；还有应用固件和源代码，用于测试和修改。

这些板支持多种功能部件，包括 LED、温度传感器、开关、扬声器、 RS-232 接口、 LCD 显示器、电位计和附加 EEPROM 存储器。

演示和开发板可用于教学环境，在实验布线区设计定制电路，从而掌握各种单片机应用。

除了 PICDEM™ 和 dsPICDEM™ 演示 / 开发板系列电路外， Microchip 还有一系列评估工具包和演示软件，适用于模拟滤波器设计、 KEELOQ® 数据安全产品 IC、CAN、 IrDA®、 PowerSmart® 电池管理、 SEEVAL® 评估系统、 Σ–Δ ADC、流速传感器，等等。

有关演示、开发和评估工具包的完整列表，请查阅 Microchip 公司网页（www.microchip.com）以及最新的 *“Product Selector Guide（产品选型指南）”*（DS00148）。

21.0 指令集综述

PIC18F1230/1330 器件具有一个含有 75 条 PIC18 内核指令的标准指令集和一个含有 8 条新指令（优化递归和软件堆栈代码）的扩展指令集。本章后面的部分将讨论该扩展指令集。

21.1 标准指令集

标准的PIC18指令集与以前的PIC®单片机指令集相比，添加了很多增强功能，并保持了易于从 PIC® 单片机指令集移植的特点。大部分指令为单字指令（16 位），只有 4 条指令是双字指令。

每条单字指令都是一个 16 位字，由操作码 （指明指令类型）和一个或多个操作数 （指定指令操作）组成。

整个指令集具有高度的正交性，并分为以下 4 种基本类型 ：

- **字节**操作类指令
- **位**操作类指令
- **立即数**操作类指令
- **控制**操作类指令

表 21-2 为 PIC18 指令集汇总，它列出了**字节**、**位**以及**立即数**和**控制**操作类指令。表21-1给出了对操作码字段的说明。

大部分**字节**操作类指令含有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 保存结果的目标寄存器 （由 “d” 指定）
3. 被访问的存储器 （由 “a” 指定）

文件寄存器指示符 “f” 指定了指令将会使用哪一个文件寄存器。目标寄存器指示符 “d” 指定了操作结果的存放位置。如果 “d” 为 0，操作结果存入 WREG 寄存器中；如果 “d” 为 1，操作结果存入指令指定的文件寄存器中。

所有**位**操作类指令都含有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 文件寄存器中的位 （由 “b” 指定）
3. 被访问的存储器 （由 “a” 指定）

位域指示符 “b” 选择操作所影响的位的编号，而文件寄存器指示符 “f” 则代表这些位所在的寄存器的编号。

立即数操作指令使用以下操作数：

- 要装入文件寄存器中的立即数 （由 “k” 指定）
- 要装入立即数的 FSR 寄存器 （由 “f” 指定）
- 不需要操作数 （由 “—” 指定）

控制操作类指令使用以下操作数：

- 程序存储器地址 （由 “n” 指定）
- `CALL` 或 `RETURN` 指令的模式 （由 “s” 指定）
- 表读和表写指令的模式 （由 “m” 指定）
- 不需要操作数 （由 “—” 指定）

除了 4 条双字指令外，所有的指令都是单字指令。双字指令将所需的信息保存在 32 位中。第二个字的高 4 位都是`1`。如果第二个字作为一条指令执行，它会执行`NOP`指令。

除非条件测试结果为 “true” 或者指令执行改变了程序计数器的值，否则执行所有的单字指令都只需要一个指令周期。对于上述两种特殊情况，执行指令需要两个指令周期，第二个指令周期执行一条 `NOP` 指令。

执行双字指令需要两个指令周期。

每个指令周期由 4 个振荡周期组成。因此，对于频率为 4 MHz 的振荡器，其正常的指令执行时间为 1 µs。如果条件测试结果为 “true” 或指令执行改变了程序计数器值，则该指令的执行时间为 2 µs。双字跳转指令（如果为 “true”）的执行需要 3 µs。

图 21-1 给出了指令的几种通用格式。所有示例均使用 “nnh” 来表示十六进制数。

指令集汇总（见表 21-2）列出了可被 Microchip MPASM™ 汇编器识别的标准指令。

第 21.1.1 节 “标准指令集”中对每条指令进行了介绍。

表 21-1： 操作码字段说明

字段	说明
a	快速操作 RAM 位 a = 0：快速操作 RAM 内的 RAM 单元 （BSR 寄存器被忽略） a = 1：由 BSR 寄存器指定的 RAM 存储区
bbb	8 位文件寄存器内的位地址 （0 到 7）。
BSR	存储区选择寄存器。用于选择当前的 RAM 存储区。
C、 DC、 Z、 OV 和 N	ALU 状态位：进位标志位、辅助进位标志位、全零标志位、溢出标志位和负标志位
d	目标寄存器选择位 d = 0：结果保存至 WREG 寄存器 d = 1：结果保存至文件寄存器 f
dest	目标寄存器：可以是 WREG 寄存器或指定的寄存器单元。
f	8 位寄存器地址 （00h 到 FFh）或 2 位 FSR 指示符 （0h 到 3h）。
f_s	12 位寄存器地址 （000h 到 FFFh）。这是源地址。
f_d	12 位寄存器地址 （000h 到 FFFh）。这是目标地址。
GIE	全局中断允许位。
k	立即数字段、常数或者标号 （可能是 8 位、 12 位或 20 位的值）。
label	标号名称。
mm * *+ *- +*	表读和表写指令的 TBLPTR 寄存器模式。 只和表读和表写指令一起使用： 不改变寄存器 （如用于表读和表写的 TBLPTR） 后增寄存器 （如用于表读和表写的 TBLPTR） 后减寄存器 （如用于表读和表写的 TBLPTR） 预增寄存器 （如用于表读和表写的 TBLPTR）
n	相对跳转指令的相对地址 （二进制补码），或 Call/Branch 和 Return 指令的直接地址。
PC	程序计数器。
PCL	程序计数器的低字节。
PCH	程序计数器的高字节。
PCLATH	程序计数器的高字节锁存器。
PCLATU	程序计数器的最高字节锁存器。
$\overline{PD}$	掉电位。
PRODH	乘积的高字节。
PRODL	乘积的低字节。
s	快速调用 / 返回模式选择位 s = 0：不对影子寄存器进行更新，也不用影子寄存器的内容更新其他寄存器 s = 1：将某些寄存器的值存入影子寄存器或把影子寄存器的值载入某些寄存器 （快速模式）
TBLPTR	21 位表指针 （指向程序存储单元）。
TABLAT	8 位表锁存器。
$\overline{TO}$	超时溢出位。
TOS	栈顶。
u	未使用或不变。
WDT	看门狗定时器。
WREG	工作寄存器 （累加器）。
x	任意值 （0 或 1）。汇编器将产生 x = 0 的代码。为了与所有的 Microchip 软件工具兼容，建议使用这种格式。
z_s	对寄存器 （源）进行间接寻址的 7 位偏移量。
z_d	对寄存器 （目标）进行间接寻址的 7 位偏移量。
{ }	可选参数。
[text]	表示变址地址。
(text)	text 的内容。
[expr]<n>	表示由指针 expr 指定的寄存器中的位 n。
→	赋值。
< >	寄存器位域。
∈	表示属于某个集合。
斜体文字	用户定义项 （字体为 Courier New）。

图 **21-1**： 指令的通用格式

面向字节的文件寄存器操作

15 … 10	9	8	7 … 0
操作码	d	a	f（寄存器地址）

d = 0 表示结果存入 WREG 寄存器
d = 1 表示结果存入文件寄存器（f）
a = 0 强制使用快速操作存储区
a = 1 使用 BSR 选择存储区
f = 8 位文件寄存器地址

指令示例：`ADDWF MYREG, W, B`

字节到字节的传送操作（双字）

15 … 12	11 … 0
操作码	f（源寄存器地址）

15 … 12	11 … 0
1111	f（目标寄存器地址）

f = 12 位文件寄存器地址

指令示例：`MOVFF MYREG1, MYREG2`

面向位的文件寄存器操作

15 … 12	11 … 9	8	7 … 0
操作码	b（位地址）	a	f（寄存器地址）

b = 占 3 位，表示文件寄存器（f）中位的位置
a = 0 强制使用快速操作存储区
a = 1 使用 BSR 选择存储区
f = 8 位文件寄存器地址

指令示例：`BSF MYREG, bit, B`

立即数操作

15 … 8	7 … 0
操作码	k（立即数）

k = 8 位立即数的值

指令示例：`MOVLW 7Fh`

控制操作

CALL、GOTO 和跳转类操作

15 … 8	7 … 0
操作码	n<7:0>（立即数）

15 … 12	11 … 0
1111	n<19:8>（立即数）

n = 20 位立即数的值

指令示例：`GOTO Label`

15 … 8	8	7 … 0
操作码	S	n<7:0>（立即数）

15 … 12	11 … 0
1111	n<19:8>（立即数）

S = 快速位

指令示例：`CALL MYFUNC`

15 … 11	10 … 0
操作码	n<10:0>（立即数）

指令示例：`BRA MYFUNC`

15 … 8	7 … 0
操作码	n<7:0>（立即数）

指令示例：`BC MYFUNC`

表 21-2： PIC18FXXXX 指令集

助记符，操作数		说明	周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
面向字节的操作类指令									
ADDWF	f, d, a	WREG 与 f 相加	1	0010	01da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
ADDWFC	f, d, a	WREG 与 f 带进位相加	1	0010	00da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
ANDWF	f, d, a	WREG 与 f 作逻辑与运算	1	0001	01da	ffff	ffff	Z 和 N	1, 2
CLRF	f, a	将 f 清零	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	将 f 取反	1	0001	11da	ffff	ffff	Z 和 N	1, 2
CPFSEQ	f, a	将 f 与 WREG 作比较，相等则跳过	1 （2 或 3）	0110	001a	ffff	ffff	无	4
CPFSGT	f, a	将 f 与 WREG 作比较，大于则跳过	1 （2 或 3）	0110	010a	ffff	ffff	无	4
CPFSLT	f, a	将 f 与 WREG 作比较，小于则跳过	1 （2 或 3）	0110	000a	ffff	ffff	无	1, 2
DECF	f, d, a	f 减 1	1	0000	01da	ffff	ffff	C、DC、Z、OV 和 N	1, 2, 3, 4
DECFSZ	f, d, a	f 减 1，为 0 则跳过	1 （2 或 3）	0010	11da	ffff	ffff	无	1, 2, 3, 4
DCFSNZ	f, d, a	f 减 1，非 0 则跳过	1 （2 或 3）	0100	11da	ffff	ffff	无	1, 2
INCF	f, d, a	f 加 1	1	0010	10da	ffff	ffff	C、DC、Z、OV 和 N	1, 2, 3, 4
INCFSZ	f, d, a	f 加 1，为 0 则跳过	1 （2 或 3）	0011	11da	ffff	ffff	无	4
INFSNZ	f, d, a	f 加 1，非 0 则跳过	1 （2 或 3）	0100	10da	ffff	ffff	无	1, 2
IORWF	f, d, a	WREG 与 f 作逻辑或运算	1	0001	00da	ffff	ffff	Z 和 N	1, 2
MOVF	f, d, a	传送 f	1	0101	00da	ffff	ffff	Z 和 N	1
MOVFF	f_s, f_d	将 f_s （源）送入第一个字 将 f_d （目标）送入第二个字	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	无	
MOVWF	f, a	将 WREG 传送给 f	1	0110	111a	ffff	ffff	无	
MULWF	f, a	WREG 乘以 f	1	0000	001a	ffff	ffff	无	1, 2
NEGF	f, a	对 f 取补	1	0110	110a	ffff	ffff	C、DC、Z、OV 和 N	
RLCF	f, d, a	f 带进位循环左移	1	0011	01da	ffff	ffff	C、Z 和 N	1, 2
RLNCF	f, d, a	f 循环左移 （不带进位）	1	0100	01da	ffff	ffff	Z 和 N	
RRCF	f, d, a	f 带进位循环右移	1	0011	00da	ffff	ffff	C、Z 和 N	
RRNCF	f, d, a	f 循环右移 （不带进位）	1	0100	00da	ffff	ffff	Z 和 N	
SETF	f, a	将 f 的内容置为全 1	1	0110	100a	ffff	ffff	无	1, 2
SUBFWB	f, d, a	WREG 减去 f （带借位）	1	0101	01da	ffff	ffff	C、DC、Z、OV 和 N	
SUBWF	f, d, a	f 减去 WREG	1	0101	11da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
SUBWFB	f, d, a	f 减去 WREG （带借位）	1	0101	10da	ffff	ffff	C、DC、Z、OV 和 N	
SWAPF	f, d, a	将 f 中的两个半字节进行交换	1	0011	10da	ffff	ffff	无	4
TSTFSZ	f, a	测试 f，为 0 则跳过	1 （2 或 3）	0110	011a	ffff	ffff	无	1, 2
XORWF	f, d, a	WREG 与 f 作逻辑异或运算	1	0001	10da	ffff	ffff	Z 和 N	

注 1： PORT 寄存器的值随端口状态的变化而不断修改 （例如， `MOVF PORTB, 1, 0`），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，虽然其数据锁存器中的值为 1，但此时外部器件将该引脚驱动为低电平，则被写回数据总线的数据值将是 0。

2： 当对 TMR0 寄存器 （以及其他适用的寄存器）执行该指令时 （并且 d = 1），如果已为其分配了预分频器，则将对该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件判断为 “true”，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 NOP 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

表 21-2： PIC18FXXXX 指令集 （续）

助记符，操作数		说明	周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
面向位的操作类指令									
BCF	f, b, a	将 f 寄存器中的某位清零	1	1001	bbba	ffff	ffff	无	1, 2
BSF	f, b, a	将 f 寄存器中的某位置 1	1	1000	bbba	ffff	ffff	无	1, 2
BTFSC	f, b, a	检测 f 中的某位，为 0 则跳过	1 （2 或 3）	1011	bbba	ffff	ffff	无	3, 4
BTFSS	f, b, a	检测 f 中的某位，为 1 则跳过	1 （2 或 3）	1010	bbba	ffff	ffff	无	3, 4
BTG	f, d, a	将 f 中的某位取反	1	0111	bbba	ffff	ffff	无	1, 2
控制操作类指令									
BC	n	进位则跳转	1 （2）	1110	0010	nnnn	nnnn	无	
BN	n	为负则跳转	1 （2）	1110	0110	nnnn	nnnn	无	
BNC	n	无进位则跳转	1 （2）	1110	0011	nnnn	nnnn	无	
BNN	n	不为负则跳转	1 （2）	1110	0111	nnnn	nnnn	无	
BNOV	n	不溢出则跳转	1 （2）	1110	0101	nnnn	nnnn	无	
BNZ	n	不为零则跳转	1 （2）	1110	0001	nnnn	nnnn	无	
BOV	n	溢出则跳转	1 （2）	1110	0100	nnnn	nnnn	无	
BRA	n	无条件跳转	2	1101	0nnn	nnnn	nnnn	无	
BZ	n	为零则跳转	1 （2）	1110	0000	nnnn	nnnn	无	
CALL	n, s	调用子程序 （第一个字）	2	1110	110s	kkkk	kkkk	无	
		（第二个字）		1111	kkkk	kkkk	kkkk		
CLRWDT	—	将看门狗定时器清零	1	0000	0000	0000	0100	$\overline{TO}$ 和 $\overline{PD}$	
DAW	—	对 WREG 进行十进制调整	1	0000	0000	0000	0111	C	
GOTO	n	跳转到地址 （第一个字）	2	1110	1111	kkkk	kkkk	无	
		（第二个字）		1111	kkkk	kkkk	kkkk		
NOP	—	空操作	1	0000	0000	0000	0000	无	
NOP	—	空操作	1	1111	xxxx	xxxx	xxxx	无	4
POP	—	弹出返回堆栈栈顶 （TOS）	1	0000	0000	0000	0110	无	
PUSH	—	压入返回堆栈栈顶 （TOS）	1	0000	0000	0000	0101	无	
RCALL	n	相对调用	2	1101	1nnn	nnnn	nnnn	无	
RESET		用软件使器件复位	1	0000	0000	1111	1111	全部	
RETFIE	s	使能中断返回	2	0000	0000	0001	000s	GIE/GIEH 和 PEIE/GIEL	
RETLW	k	返回时将立即数送入 WREG	2	0000	1100	kkkk	kkkk	无	
RETURN	s	从子程序返回	2	0000	0000	0001	001s	无	
SLEEP	—	进入待机模式	1	0000	0000	0000	0011	$\overline{TO}$ 和 $\overline{PD}$	

注 **1：** PORT 寄存器的值随端口状态的变化而不断修改（例如， `MOVF PORTB, 1, 0`），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，虽然其数据锁存器中的值为 1，但此时外部器件将该引脚驱动为低电平，则被写回数据总线的数据值将是 0。

2： 当对 TMR0 寄存器 （以及其他适用的寄存器）执行该指令时 （并且 d = 1），如果已为其分配了预分频器，则将对该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件判断为 “true”，则该指令需要两个周期。第二个周期执行一条 `NOP` 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 `NOP` 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

表 21-2： PIC18FXXXX 指令集 （续）

助记符，操作数		说明	周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
立即数操作类指令									
ADDLW	k	WREG 与立即数相加	1	0000	1111	kkkk	kkkk	C、DC、Z、OV 和 N	
ANDLW	k	立即数与 WREG 作逻辑与运算	1	0000	1011	kkkk	kkkk	Z 和 N	
IORLW	k	立即数与 WREG 作逻辑或运算	1	0000	1001	kkkk	kkkk	Z 和 N	
LFSR	f, k	将 12 位立即数第二个字	2	1110	1110	00ff	kkkk	无	
		传送 到 FSR(f) 第一个字		1111	0000	kkkk	kkkk		
MOVLB	k	将立即数送入 BSR<3:0>	1	0000	0001	0000	kkkk		
MOVLW	k	将立即数送入 WREG	1	0000	1110	kkkk	kkkk	无	
MULLW	k	立即数与 WREG 相乘	1	0000	1101	kkkk	kkkk	无	
RETLW	k	返回时将立即数送入 WREG	2	0000	1100	kkkk	kkkk	无	
SUBLW	k	立即数减去 WREG	1	0000	1000	kkkk	kkkk	无	
								C、DC、Z、OV 和 N	
XORLW	k	立即数与 WREG 作逻辑异或运算	1	0000	1010	kkkk	kkkk	Z 和 N	
数据存储器 ↔ 程序存储器操作类指令									
TBLRD*		表读	2	0000	0000	0000	1000	无	
TBLRD*+		后增表读		0000	0000	0000	1001	无	
TBLRD*-		后减表读		0000	0000	0000	1010	无	
TBLRD+*		预增表读		0000	0000	0000	1011	无	
TBLWT*		表写	2	0000	0000	0000	1100	无	
TBLWT*+		后增表写		0000	0000	0000	1101	无	
TBLWT*-		后减表写		0000	0000	0000	1110	无	
TBLWT+*		预增表写		0000	0000	0000	1111	无	

注 1： PORT 寄存器的值随端口状态的变化而不断修改 （例如， `MOVF PORTB, 1, 0`），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，虽然其数据锁存器中的值为 `1`，但此时外部器件将该引脚驱动为低电平，则被写回数据总线的数据值将是 `0`。

2： 当对 TMR0 寄存器 （以及其他适用的寄存器）执行该指令时 （并且 d = `1`），如果已为其分配了预分频器，则将对该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件判断为 “true”，则该指令需要两个周期。第二个周期执行一条 `NOP` 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 `NOP` 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

21.1.1 标准指令集

ADDLW **W 与立即数相加**

语法： ADDLW k

操作数： 0 ≤ k ≤ 255

操作： (W) + k → W

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0000	1111	kkkk	kkkk

说明： 将 W 的内容与 8 位立即数 k 相加，结果保存在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例： `ADDLW 15h`

执行指令前
W = 10h
执行指令后
W = 25h

ADDWF **W 与 f 寄存器相加**

语法： ADDWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (W) + (f) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0010	01da	ffff	ffff

说明： 将 W 的内容与 f 寄存器的内容相加。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例： `ADDWF REG, 0, 0`

执行指令前
W = 17h
REG = 0C2h
执行指令后
W = 0D9h
REG = 0C2h

注：	所有的 PIC18 指令都可能在其指令助记符之前使用标号参数，用于符号寻址。如果使用了标号，那么指令语法将变为：{ 标号 } 指令参数。

ADDWFC　W 与 f 带进位相加

语法：	ADDWFC　f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(W) + (f) + (C) → dest
受影响的状态位：	N、 OV、 C、 DC 和 Z
机器码：	0010　00da　ffff　ffff
说明：	将 W 的内容、进位标志位与寄存器 f 的内容相加。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存储在寄存器 f 中。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例：　ADDWFC　REG, 0, 1

执行指令前
- 进位标志位 = 1
- REG = 02h
- W = 4Dh

执行指令后
- 进位标志位 = 0
- REG = 02h
- W = 50h

ANDLW　立即数与 W 寄存器作逻辑与运算

语法：	ANDLW　k
操作数：	0 ≤ k ≤ 255
操作：	(W) .AND. k → W
受影响的状态位：	N 和 Z
机器码：	0000　1011　kkkk　kkkk
说明：	将 W 的内容与 8 位立即数 k 作逻辑与运算。结果保存在 W 寄存器中。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例：　ANDLW　05Fh

执行指令前
- W = A3h

执行指令后
- W = 03h

ANDWF 将 W 与 f 作逻辑与运算

语法： ANDWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (W) .AND.(f) → dest

受影响的状态位： N 和 Z

机器码：

0001	01da	ffff	ffff

说明： 将 W 的内容与寄存器 f 的内容作逻辑与运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： ANDWF REG, 0, 0

执行指令前
W = 17h
REG = C2h
执行指令后
W = 02h
REG = C2h

BC 进位则跳转

语法： BC n

操作数： -128 ≤ n ≤ 127

操作： 如果进位标志位为 1
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0010	nnnn	nnnn

说明： 如果进位标志位为 1，那么程序将跳转。二进制补码 "2n" 与 PC 相加。因为 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数： 1

指令周期数： 1 （2）

Q 周期操作：
如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BC 5

执行指令前
PC = 地址 （HERE）
执行指令后
如果进位标志位 = 1；
PC = 地址 （HERE + 12）
如果进位标志位 = 0；
PC = 地址 （HERE + 2）

BCF 将 f 寄存器中的某位清零

语法： BCF f, b {,a}

操作数： 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

操作： 0 → f<b>

受影响的状态位： 无

机器码：

1001	bbba	ffff	ffff

说明： 将寄存器 f 中的位 b 清零。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： BCF FLAG_REG, 7, 0

执行指令前
FLAG_REG = C7h
执行指令后
FLAG_REG = 47h

BN 为负则跳转

语法： BN n

操作数： -128 ≤ n ≤ 127

操作： 如果负标志位为 1
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0110	nnnn	nnnn

说明： 如果负标志位为 1，那么程序将跳转。二进制补码 "2n" 与 PC 相加。因为 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：
如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BN Jump

执行指令前
PC = 地址 （HERE）
执行指令后
如果负标志位 = 1；
PC = 地址 （Jump）
如果负标志位 = 0；
PC = 地址 （HERE + 2）

BNC　无进位则跳转

语法：BNC　n

操作数：-128 ≤ n ≤ 127

操作：如果进位标志位为 0
(PC) + 2 + 2n → PC

受影响的状态位：无

机器码：

1110	0011	nnnn	nnnn

说明：如果进位标志位为 0，那么程序将跳转。二进制补码 “2n” 与 PC 相加。因为 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数：1

指令周期数：1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：

```
HERE      BNC  Jump
```

执行指令前
PC　=　地址 （HERE）
执行指令后
如果进位标志位 =　0；
PC　=　地址 （Jump）
如果进位标志位 =　1；
PC　=　地址 （HERE + 2）

BNN　不为负则跳转

语法：BNN　n

操作数：-128 ≤ n ≤ 127

操作：如果负标志位为 0
(PC) + 2 + 2n → PC

受影响的状态位：无

机器码：

1110	0111	nnnn	nnnn

说明：如果负标志位为 0，那么程序将跳转。二进制补码 “2n” 与 PC 相加。因为 PC 要先递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数：1

指令周期数：1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：

```
HERE      BNN  Jump
```

执行指令前
PC　=　地址 （HERE）
执行指令后
如果负标志位 =　0；
PC　=　地址 （Jump）
如果负标志位 =　1；
PC　=　地址 （HERE + 2）

BNOV 不溢出则跳转

语法： BNOV n

操作数： -128 ≤ n ≤ 127

操作： 如果溢出标志位为 0
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0101	nnnn	nnnn

说明： 如果溢出标志位为 0，那么程序将跳转。二进制补码 “2n” 与 PC 相加。因为 PC 要先递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BNOV Jump

执行指令前
PC = 地址 （HERE）

执行指令后
如果溢出标志位 = 0；
PC = 地址 （Jump）
如果溢出标志位 = 1；
PC = 地址 （HERE + 2）

BNZ 不为零则跳转

语法： BNZ n

操作数： -128 ≤ n ≤ 127

操作： 如果全零标志位为 0
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0001	nnnn	nnnn

说明： 如果全零标志位为 0，那么程序将跳转。二进制补码 “2n” 与 PC 相加。因为 PC 要先递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BNZ Jump

执行指令前
PC = 地址 （HERE）

执行指令后
如果全零标志位= 0；
PC = 地址 （Jump）
如果全零标志位= 1；
PC = 地址 （HERE + 2）

BRA 无条件跳转

语法： BRA n

操作数： $-1024 \leqslant n \leqslant 1023$

操作： $(PC) + 2 + 2n \rightarrow PC$

受影响的状态位： 无

机器码：

1101	0nnn	nnnn	nnnn

说明： 将二进制补码 “2n” 与 PC 相加。因为 PC 要先递增才能取下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例： `HERE BRA Jump`

执行指令前
PC = 地址 （HERE）
执行指令后
PC = 地址 （Jump）

BSF 将 f 寄存器中的某位置 1

语法： BSF f, b {,a}

操作数： $0 \leqslant f \leqslant 255$
$0 \leqslant b \leqslant 7$
$a \in [0,1]$

操作： 1 → f<b>

受影响的状态位： 无

机器码：

1000	bbba	ffff	ffff

说明： 将寄存器 f 中的位 b 置 1。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： `BSF FLAG_REG, 7,1`

执行指令前
FLAG_REG = 0Ah
执行指令后
FLAG_REG = 8Ah

BTFSC 测试寄存器中的位，为 0 则跳过

语法： BTFSC f, b {,a}

操作数： 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

操作： 如果 （f<b>） = 0 则跳过

受影响的状态位： 无

机器码：

1011	bbba	ffff	ffff

说明： 如果寄存器 f 中的位 b 为 0，则跳过下一条指令。即在位 b 为 0 时，丢弃在当前指令执行期间取到的下一条指令，转而执行一条 NOP 指令，使该指令变成双周期指令。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1 （2）
注：如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址 （HERE）
执行指令后
如果 FLAG<1> = 0；
PC = 地址 （TRUE）
如果 FLAG<1> = 1；
PC = 地址 （FALSE）

BTFSS 测试寄存器中的位，为 1 则跳过

语法： BTFSS f, b {,a}

操作数： 0 ≤ f ≤ 255
0 ≤ b < 7
a ∈ [0,1]

操作： 如果 （f<b>） = 1 则跳过

受影响的状态位： 无

机器码：

1010	bbba	ffff	ffff

说明： 如果寄存器 f 中的位 b 为 1，则跳过下一条指令。即在位 b 为 1 时，丢弃在当前指令执行期间取到的下一条指令，转而执行一条 NOP 指令，使该指令变成双周期指令。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1 （2）
注：如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址 （HERE）
执行指令后
如果 FLAG<1> = 0；
PC = 地址 （FALSE）
如果 FLAG<1> = 1；
PC = 地址 （TRUE）

BTG 将 f 中的某位取反

语法： BTG f, b {,a}

操作数： $0 \leqslant f \leqslant 255$
$0 \leqslant b < 7$
$a \in [0,1]$

操作： $(\overline{f<b>}) \rightarrow f<b>$

受影响的状态位： 无

机器码：

0111	bbba	ffff	ffff

说明： 将数据存储单元 f 中的位 b 取反。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： BTG PORTC, 4, 0

执行指令前：
PORTC = 0111 0101 [75h]
执行指令后：
PORTC = 0110 0101 [65h]

BOV 溢出则跳转

语法： BOV n

操作数： $-128 \leqslant n \leqslant 127$

操作： 如果溢出标志位为 1
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0100	nnnn	nnnn

说明： 如果溢出标志位为 1，那么程序将跳转。二进制补码 "2n" 与 PC 相加。因为 PC 要先递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：
如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BOV Jump

执行指令前
PC = 地址 （HERE）
执行指令后
如果溢出标志位 = 1；
PC = 地址 （Jump）
如果溢出标志位 = 0；
PC = 地址 （HERE + 2）

BZ 为零则跳转

语法：	BZ n
操作数：	-128 ≤ n ≤ 127
操作：	如果全零标志位为 1 (PC) + 2 + 2n → PC
受影响的状态位：	无
机器码：	1110 | 0000 | nnnn | nnnn
说明：	如果全零标志位为 1，那么程序将跳转。二进制补码 “2n”与 PC 相加。因为 PC 要先递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。这种情况下，该指令是一条双周期指令。
指令字数：	1
指令周期数：	1 （2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BZ Jump

执行指令前
PC = 地址 （HERE）
执行指令后
如果全零标志位= 1；
PC = 地址（Jump）
如果全零标志位= 0；
PC = 地址 （HERE + 2）

CALL 调用子程序

语法：	CALL k {,s}
操作数：	0 ≤ k ≤ 1048575 s ∈ [0,1]
操作：	(PC)+4 → TOS， k → PC<20:1>, 如果 s = 1 (W) → WS, (STATUS) → STATUSS, (BSR) → BSRS
受影响的状态位：	无

机器码：				
第一个字 （k<7:0>）	1110	110s	$k_7$kkk	kkkk$_0$
第二个字 （k<19:8>）	1111	k_{19}kkk	kkkk	kkkk$_8$

说明：	可在整个 2 MB 的存储器范围内进行子程序调用。首先，将返回地址 （PC+4）压入返回堆栈。如果 s = 1，还会将 W、STATUS 和 BSR 寄存器的内容存入它们各自的影子寄存器 WS、STATUSS 和 BSRS。如果 s = 0，将不会进行任何更新 （默认）。然后将 “k”的 20 位值装入 PC<20:1>。 CALL 是一条双周期指令。
指令字数：	2
指令周期数：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>	将 PC 压入堆栈	读立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例： HERE CALL THERE,1

执行指令前
PC = 地址 （HERE）
执行指令后
PC = 地址 （THERE）
TOS = 地址 （HERE + 4）
WS = W
BSRS = BSR
STATUSS = STATUS

CLRF 将 f 清零

语法： CLRF f {,a}

操作数： $0 \leqslant f \leqslant 255$
$a \in [0,1]$

操作： 000h → f
1 → Z

受影响的状态位： Z

机器码：

0110	101a	ffff	ffff

说明： 清零指定寄存器的内容。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： CLRF FLAG_REG,1

执行指令前
FLAG_REG = 5Ah
执行指令后
FLAG_REG = 00h

CLRWDT 将看门狗定时器清零

语法： CLRWDT

操作数： 无

操作： 000h → WDT，
000h → WDT 后分频器，
1 → $\overline{TO}$，
1 → $\overline{PD}$

受影响的状态位： $\overline{TO}$ 和 $\overline{PD}$

机器码：

0000	0000	0000	0100

说明： CLRWDT 指令复位看门狗定时器，而且还会复位 WDT 的后分频器。状态位 $\overline{TO}$ 和 $\overline{PD}$ 被置 1。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	空操作

示例： CLRWDT

执行指令前
WDT 计数器 = ?
执行指令后
WDT 计数器 = 00h
WDT 后分频器 = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF 将 f 取反

语法：	COMF f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	$(\bar{f})$ → dest
受影响的状态位：	N 和 Z
机器码：	0001 | 11da | ffff | ffff
说明：	将寄存器 f 的内容取反。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例： `COMF REG, 0, 0`

执行指令前
REG = 13h

执行指令后
REG = 13h
W = ECh

CPFSEQ 比较 f 和 W，如果 f = W 则跳过

语法：	CPFSEQ f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(f) – (W), 如果 (f) = (W) 则跳过 （无符号比较）
受影响的状态位：	无
机器码：	0110 | 001a | ffff | ffff
说明：	通过执行无符号减法，将数据存储单元 f 的内容与 W 的内容作比较。 如果 f = W，则所取的指令被丢弃，转而执行一条 NOP 指令，从而使该指令变成双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。
指令字数：	1
指令周期数：	1 （2） **注：** 如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE      CPFSEQ REG, 0
NEQUAL    :
EQUAL     :
```

执行指令前
PC 地址 = HERE
W = ?
REG = ?

执行指令后
如果 REG = W；
PC = 地址 （EQUAL）
如果 REG ≠ W；
PC = 地址 （NEQUAL）

 超前信息

CPFSGT 比较 f 和 W，如果 f > W 则跳过

语法：	CPFSGT f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(f) – (W), 如果 (f) > (W) 则跳过 （无符号比较）
受影响的状态位：	无
机器码：	0110 010a ffff ffff
说明：	通过执行无符号减法，将数据存储单元 f 的内容与 W 的内容作比较。 如果 f 中的数值大于 WREG 中的数值，则所取的指令会被丢弃，转而执行一条 NOP 指令，从而使该指令变成双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1（2） **注：** 如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE       CPFSGT REG, 0
NGREATER   :
GREATER    :
```

执行指令前
PC = 地址（HERE）
W = ?

执行指令后
如果 REG > W；
PC = 地址（GREATER）
如果 REG ≤ W；
PC = 地址（NGREATER）

CPFSLT 比较 f 和 W，如果 f < W 则跳过

语法：	CPFSLT f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(f) – (W), 如果 (f) < (W) 则跳过 （无符号比较）
受影响的状态位：	无
机器码：	0110 000a ffff ffff
说明：	通过执行无符号减法，将数据存储单元 f 的内容与 W 的内容作比较。 如果 f 中的数值小于 W 中的数值，则所取的指令被丢弃，转而执行一条 NOP 指令，使该指令变成双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
指令字数：	1
指令周期数：	1（2） **注：** 如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

执行指令前
PC = 地址（HERE）
W = ?

执行指令后
如果 REG < W；
PC = 地址（LESS）
如果 REG ≥ W；
PC = 地址（NLESS）

DAW 对 W 寄存器进行十进制调整

语法： DAW

操作数： 无

操作：
如果 [W<3:0> > 9] 或 [DC = 1] 那么
(W<3:0>) + 6 → W<3:0>；
否则
(W<3:0>) → W<3:0>

如果 [W<7:4> + DC > 9] 或 [C = 1] 那么
(W<7:4>) + 6 + DC → W<7:4>；
否则
(W<7:4>) + DC → W<7:4>

受影响的状态位： C

机器码：

0000	0000	0000	0111

说明： DAW 指令调整 W 内的 8 位数值，即前两个压缩 BCD 格式的变量之和，并产生一个正确的压缩 BCD 格式结果。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 W	处理数据	写 寄存器 W

例 1： DAW

执行指令前
W = A5h
C = 0
DC = 0
执行指令后
W = 05h
C = 1
DC = 0

例 2：

执行指令前
W = CEh
C = 0
DC = 0
执行指令后
W = 34h
C = 1
DC = 0

DECF f 减 1

语法： DECF f {,d {,a}}

操作数：
0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) – 1 → dest

受影响的状态位： C、 DC、 N、 OV 和 Z

机器码：

0000	01da	ffff	ffff

说明： 将寄存器 f 的内容减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例： DECF CNT, 1, 0

执行指令前
CNT = 01h
Z = 0
执行指令后
CNT = 00h
Z = 1

DECFSZ　f 减 1，为 0 则跳过

语法：	DECFSZ　f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f) – 1 → dest， 结果为 0 时跳过
受影响的状态位：	无
机器码：	0010　11da　ffff　ffff
说明：	将寄存器 f 的内容减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果为 0，则丢弃已取的下一条指令，转而执行一条 NOP 指令，使该指令成为双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1（2） **注：**如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE      DECFSZ   CNT, 1, 1
          GOTO     LOOP
CONTINUE
```

执行指令前
　PC = 地址 （HERE）

执行指令后
　CNT = CNT – 1
　如果 CNT = 0；
　　PC = 地址 （CONTINUE）
　如果 CNT ≠ 0；
　　PC = 地址 （HERE + 2）

DCFSNZ　f 减 1，非 0 则跳过

语法：	DCFSNZ　f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f) – 1 → dest， 结果 ≠ 0 时跳过
受影响的状态位：	无
机器码：	0100　11da　ffff　ffff
说明：	将寄存器 f 的内容减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果不为 0，则丢弃已经取到的下一条指令，转而执行一条 NOP 指令，使该指令成为双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1（2） **注：**如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE     DCFSNZ  TEMP, 1, 0
ZERO     :
NZERO    :
```

执行指令前
　TEMP = ?

执行指令后
　TEMP = TEMP – 1
　如果 TEMP = 0；
　　PC = 地址 （ZERO）
　如果 TEMP ≠ 0；
　　PC = 地址 （NZERO）

GOTO 无条件跳转

语法： GOTO k

操作数： 0 ≤ k ≤ 1048575

操作： k → PC<20:1>

受影响的状态位： 无

机器码：

第一个字（k<7:0>）	1110	1111	$k_7$kkk	kkkk$_0$
第二个字（k<19:8>）	1111	k_{19}kkk	kkkk	kkkk$_8$

说明： GOTO 指令允许无条件跳转到整个 2 MB 存储器范围中的任何位置。将 20 位值 k 装入 PC<20:1>。 GOTO 始终为一条双周期指令。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>	空操作	读立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例： GOTO THERE

执行指令后
PC = 地址（THERE）

INCF f 加 1

语法： INCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) + 1 → dest

受影响的状态位： C、 DC、 N、 OV 和 Z

机器码：

0010	10da	ffff	ffff

说明： 将寄存器 f 的内容加 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： INCF CNT, 1, 0

执行指令前
CNT = FFh
Z = 0
C = ?
DC = ?

执行指令后
CNT = 00h
Z = 1
C = 1
DC = 1

INCFSZ　　f 加 1，为 0 则跳过

语法：　INCFSZ　f {,d {,a}}

操作数：　0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作：　(f) + 1 → dest
结果为 0 时跳过

受影响的状态位：　无

机器码：

0011	11da	ffff	ffff

说明：　将寄存器 f 的内容加 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果结果为 0，则丢弃已取的下一条指令，转而执行一条 NOP 指令，使该指令成为双周期指令。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数：　1

指令周期数：　1（2）
注：如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

执行指令前
PC = 地址（HERE）
执行指令后
CNT = CNT + 1
如果 CNT = 0；
PC = 地址（ZERO）
如果 CNT ≠ 0；
PC = 地址（NZERO）

INFSNZ　　f 加 1，非 0 则跳过

语法：　INFSNZ　f {,d {,a}}

操作数：　0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作：　(f) + 1 → dest
结果 ≠ 0 时跳过

受影响的状态位：　无

机器码：

0011	11da	ffff	ffff

说明：　将寄存器 f 的内容加 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果结果不为 0，则丢弃已经取到的下一条指令，转而执行一条 NOP 指令，使该指令成为双周期指令。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数：　1

指令周期数：　1（2）
注：如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

执行指令前
PC = 地址（HERE）
执行指令后
REG = REG + 1
如果 REG ≠ 0；
PC = 地址（NZERO）
如果 REG = 0；
PC = 地址（ZERO）

IORLW 立即数与 W 作逻辑或运算

语法：	IORLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .OR. k → W
受影响的状态位：	N 和 Z
机器码：	0000 1001 kkkk kkkk
说明：	将 W 的内容与 8 位立即数 k 作逻辑或运算。结果保存在 W 寄存器中。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入 寄存器 W

示例： `IORLW 35h`

执行指令前
W = 9Ah
执行指令后
W = BFh

IORWF 将 W 与 f 作逻辑或运算

语法：	IORWF f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(W) .OR. (f) → dest
受影响的状态位：	N 和 Z
机器码：	0001 00da ffff ffff
说明：	将 W 与寄存器 f 的内容作逻辑或运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例： `IORWF RESULT, 0, 1`

执行指令前
RESULT = 13h
W = 91h
执行指令后
RESULT = 13h
W = 93h

LFSR 装入 FSR

语法：	LFSR f, k
操作数：	0 ≤ f ≤ 2 0 ≤ k ≤ 4095
操作：	k → FSRf
受影响的状态位：	无

机器码：

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

说明： 将 12 位立即数 k 装入由 f 指向的指针寄存器。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k 的 MSB	处理数据	写立即数 k 的 MSB 到 FSRfH
译码	读立即数 k 的 LSB	处理数据	将立即数 k 写入 FSRfL

示例： `LFSR 2, 3ABh`

执行指令后
FSR2H = 03h
FSR2L = ABh

MOVF 传送 f

语法：	MOVF f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	f → dest
受影响的状态位：	N 和 Z

机器码：

0101	00da	ffff	ffff

说明： 根据 d 的状态，将寄存器 f 的内容送入目标单元。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。f 可以为 256 字节存储区中的任何单元。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写 W

示例： `MOVF REG, 0, 0`

执行指令前
REG = 22h
W = FFh
执行指令后
REG = 22h
W = 22h

MOVFF 将源寄存器的内容送入目标寄存器

语法： MOVFF f_s,f_d

操作数： $0 \leqslant f_s \leqslant 4095$
$0 \leqslant f_d \leqslant 4095$

操作： $(f_s) \rightarrow f_d$

受影响的状态位： 无

机器码：

第一个字 （源）	1100	ffff	ffff	$ffff_s$
第二个字 （目标）	1111	ffff	ffff	$ffff_d$

说明： 将源寄存器 f_s 的内容送入目标寄存器 f_d。
源寄存器 f_s 可以是 4096 字节数据空间（000h 到 FFFh）中的任何单元，目标寄存器 f_d 也可以是 000h 到 FFFh 中的任何单元。
源或目标寄存器都可以是 W （这是个有用的特例）。
MOVFF 指令对于将数据存储单元中的内容送入外设寄存器 （如发送缓冲器或 I/O 端口）的场合非常有用。
MOVFF 指令中不能使用 PCL、 TOSU、TOSH 或 TOSL 作为目标寄存器。

指令字数： 2

指令周期数： 2（3）

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f（源寄存器）	处理数据	空操作
译码	空操作 非无效读取	空操作	写 目标寄存器 f

示例： MOVFF REG1, REG2

执行指令前
REG1 = 33h
REG2 = 11h
执行指令后
REG1 = 33h
REG2 = 33h

MOVLB 将立即数送入 BSR 的低半字节

语法： MOVLW k

操作数： $0 \leqslant k \leqslant 255$

操作： $k \rightarrow BSR$

受影响的状态位： 无

机器码：

0000	0001	kkkk	kkkk

说明： 将 8 位立即数 k 装入存储区选择寄存器（BSR）。不管 k_7:k_4 的值如何，BSR<7:4> 的值将始终保持为 0。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	将立即数 k 写入 BSR

示例： MOVLB 5

执行指令前
BSR 寄存器 = 02h
执行指令后
BSR 寄存器 = 05h

MOVLW 将立即数送入 **W**

语法： MOVLW k

操作数： 0 ≤ k ≤ 255

操作： k → W

受影响的状态位： 无

机器码：

0000	1110	kkkk	kkkk

说明： 将 8 位立即数 k 装入 W。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入寄存器 W

示例： `MOVLW 5Ah`

执行指令后
W = 5Ah

MOVWF 将 **W** 的内容送入 **f**

语法： MOVWF f {,a}

操作数： 0 ≤ f ≤ 255
a ∈ [0,1]

操作： (W) → f

受影响的状态位： 无

机器码：

0110	111a	ffff	ffff

说明： 将 W 寄存器中的数据送入寄存器 f。f 可以是 256 字节存储区中的任何单元。如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写 寄存器 f

示例： `MOVWF REG, 0`

执行指令前
W = 4Fh
REG = FFh

执行指令后
W = 4Fh
REG = 4Fh

MULLW 将立即数与 W 的内容相乘

语法：	MULLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) x k → PRODH:PRODL
受影响的状态位：	无
机器码：	0000 1101 kkkk kkkk
说明：	将 W 的内容与 8 位立即数 k 进行无符号的乘法运算。 16 位的结果存储在 PRODH:PRODL 寄存器对中。其中 PRODH 存储高字节。 W 的内容不变。 所有状态标志位都不受影响。 请注意此操作不可能发生溢出或进位。结果有可能为零，但不会反映到相应的标志位。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写寄存器 PRODH: PRODL

示例： `MULLW 0C4h`

执行指令前
W = E2h
PRODH = ?
PRODL = ?

执行指令后
W = E2h
PRODH = ADh
PRODL = 08h

MULWF 将 W 与 f 的内容相乘

语法：	MULWF f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(W) x (f) → PRODH:PRODL
受影响的状态位：	无
机器码：	0000 001a ffff ffff
说明：	将 W 的内容与寄存器单元 f 的内容执行无符号的乘法运算。运算的 16 位结果存储在 PRODH:PRODL 寄存器对中。其中 PRODH 存储高字节。 W 和 f 的内容都不变。 所有状态标志位都不受影响。 请注意此操作不可能发生溢出或进位。结果有可能为零，但不会反映到相应的标志位。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写寄存器 PRODH: PRODL

示例： `MULWF REG, 1`

执行指令前
W = C4h
REG = B5h
PRODH = ?
PRODL = ?

执行指令后
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

NEGF 对 f 取补

语法：	NEGF f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	$(\bar{f}) + 1 \rightarrow f$
受影响的状态位：	N、 OV、 C、 DC 和 Z
机器码：	0110 | 110a | ffff | ffff
说明：	用二进制补码对单元 f 取补。结果存储在数据存储单元 f 中。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写 寄存器 f

示例： `NEGF REG, 1`

执行指令前
REG = 0011 1010 [3Ah]
执行指令后
REG = 1100 0110 [C6h]

NOP 空操作

语法：	NOP
操作数：	无
操作：	空操作
受影响的状态位：	无
机器码：	0000 | 0000 | 0000 | 0000 1111 | xxxx | xxxx | xxxx
说明：	不执行任何操作。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作

示例：

无。

POP 弹出返回堆栈栈顶的内容

语法：	POP
操作数：	无
操作：	(TOS) → 丢弃
受影响的状态位：	无
机器码：	0000 0000 0000 0110
说明：	从返回堆栈弹出 TOS 值并丢弃。然后，前一个压入返回堆栈的值成为 TOS 值。此指令可以让用户正确管理返回堆栈，从而实现软件堆栈。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	弹出 TOS 值	空操作

示例：

```
POP
GOTO    NEW
```

执行指令前

TOS	=	0031A2h
堆栈（下 1 级）	=	014332h

执行指令后

TOS	=	014332h
PC	=	NEW

PUSH 将数据压入返回堆栈栈顶

语法：	PUSH
操作数：	无
操作：	(PC + 2) → TOS
受影响的状态位：	无
机器码：	0000 0000 0000 0101
说明：	PC + 2 的值被压入返回堆栈的栈顶。原先的 TOS 值被压入堆栈的下一级。此指令允许通过修改 TOS 并将其压入返回堆栈来实现软件堆栈。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	将 PC+2 压入返回堆栈	空操作	空操作

示例：

```
PUSH
```

执行指令前

TOS	=	345Ah
PC	=	0124h

执行指令后

PC	=	0126h
TOS	=	0126h
堆栈（下一级）	=	345Ah

RCALL 相对调用

语法： RCALL n

操作数： -1024 ≤ n ≤ 1023

操作： (PC) + 2 → TOS，
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1101	1nnn	nnnn	nnnn

说明： 从当前地址跳转 （最多 1 KB）来调用子程序。首先，将返回地址 （PC + 2）压入返回堆栈。然后，将二进制补码 “2n”与 PC 相加。因为 PC 要先递增才能取下一条指令，因此新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 n 将 PC 压入堆栈	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例： HERE RCALL Jump

执行指令前
PC = 地址 （HERE）
执行指令后
PC = 地址 （Jump）
TOS = 地址 （HERE + 2）

RESET 复位

语法： RESET

操作数： 无

操作： 将所有受 $\overline{MCLR}$ 复位影响的寄存器和标志位复位。

受影响的状态位： 全部

机器码：

0000	0000	1111	1111

说明： 此指令可提供一种用软件实现 $\overline{MCLR}$ 复位的方法。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	开始复位	空操作	空操作

示例： RESET

执行指令后
寄存器 = 复位值
标志位 * = 复位值

RETFIE　从中断返回

语法：	RETFIE {s}
操作数：	s ∈ [0,1]
操作：	(TOS) → PC， 1 → GIE/GIEH 或 PEIE/GIEL， 如果 s = 1 (WS) → W， (STATUSS) → STATUS 寄存器， (BSRS) → BSR， PCLATU 和 PCLATH 保持不变。
受影响的状态位：	GIE/GIEH 和 PEIE/GIEL。
机器码：	0000 0000 0001 000s
说明：	从中断返回。执行出栈操作，将栈顶（TOS）的内容装入 PC。通过将高或低优先级全局中断允许位置 1，来允许中断。如果 s = 1，则影子寄存器 WS、STATUSS 和 BSRS 的内容将被装入对应的寄存器 W、Status 和 BSR。如果 s = 0，则不更新这些寄存器（默认）。
指令字数：	1
指令周期数：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	从堆栈弹出 PC 值 将 GIEH 或 GIEL 置 1
空操作	空操作	空操作	空操作

示例：　RETFIE 1

中断后

PC	=	TOS
W	=	WS
BSR	=	BSRS
STATUS	=	STATUSS
GIE/GIEH, PEIE/GIEL	=	1

RETLW　返回时将立即数送入 W

语法：	RETLW k
操作数：	0 ≤ k ≤ 255
操作：	k → W， (TOS) → PC， PCLATU 和 PCLATH 保持不变。
受影响的状态位：	无
机器码：	0000 1100 kkkk kkkk
说明：	将 8 位立即数 k 装入 W。将栈顶内容（返回地址）装入程序计数器。高位地址锁存器（PCLATH）的内容保持不变。
指令字数：	1
指令周期数：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	从堆栈弹出 PC 值，写入 W
空操作	空操作	空操作	空操作

示例：

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
   :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
   :
   :
    RETLW kn   ; End of table
```

执行指令前

W = 07h

执行指令后

W = kn 的值

RETURN 从子程序返回

语法：	RETURN {s}
操作数：	s ∈ [0,1]
操作：	(TOS) → PC， 如果 s = 1 (WS) → W， (STATUSS) → STATUS 寄存器， (BSRS) → BSR， PCLATU 和 PCLATH 保持不变。
受影响的状态位：	无
机器码：	0000 0000 0001 001s
说明：	从子程序返回。执行出栈操作，将栈顶（TOS）内容装入程序计数器。如果 s = 1，将影子寄存器 WS、 STATUSS 和 BSRS 的内容装入相应的 W、 STATUS 和 BSR 寄存器。如果 s = 0，则不更新这些寄存器 （默认）。
指令字数：	1
指令周期数：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	从堆栈弹出 PC 值
空操作	空操作	空操作	空操作

示例： RETURN

执行指令后：
PC = TOS

RLCF f 带进位循环左移

语法：	RLCF f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f<n>) → dest<n+1>， (f<7>) → C， (C) → dest<0>
受影响的状态位：	C、 N 和 Z
机器码：	0011 01da ffff ffff
说明：	将寄存器 f 的内容连同进位标志位一起循环左移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。

C ← 寄存器 f ←

指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： RLCF REG, 0, 0

执行指令前
REG = 1110 0110
C = 0

执行指令后
REG = 1110 0110
W = 1100 1100
C = 1

RLNCF　f 循环左移 （不带进位）

语法：　RLNCF　f {,d {,a}}

操作数：　0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作：　(f<n>) → dest<n+1>，
(f<n>) → dest<0>

受影响的状态位：　N 和 Z

机器码：

0100	01da	ffff	ffff

说明：　将寄存器 f 的内容循环左移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

寄存器 f

指令字数：　1

指令周期数：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例：　RLNCF　REG, 1, 0

执行指令前
REG　=　1010 1011

执行指令后
REG　=　0101 0111

RRCF　f 带进位循环右移

语法：　RRCF　f {,d {,a}}

操作数：　0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作：　(f<n>) → dest<n-1>，
(f<0>) → C，
(C) → dest<7>

受影响的状态位：　C、 N 和 Z

机器码：

0011	00da	ffff	ffff

说明：　将寄存器 f 的内容连同进位标志位一起循环右移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

C　寄存器 f

指令字数：　1

指令周期数：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例：　RRCF　REG, 0, 0

执行指令前
REG　=　1110 0110
C　=　0

执行指令后
REG　=　1110 0110
W　=　0111 0011
C　=　0

RRNCF　f 循环右移 （不带进位）

语法： RRNCF　f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f<n>) → dest<n-1>，
(f<0>) → dest<7>

受影响的状态位： N 和 Z

机器码：

0100	00da	ffff	ffff

说明： 将寄存器 f 的内容循环右移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区，忽略 BSR 的值。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

寄存器 f

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

例 1：　RRNCF　REG, 1, 0

执行指令前
REG = 1101 0111
执行指令后
REG = 1110 1011

例 2：　RRNCF　REG, 0, 0

执行指令前
W = ?
REG = 1101 0111
执行指令后
W = 1110 1011
REG = 1101 0111

SETF　将 f 的内容置为全 1

语法： SETF　f {,a}

操作数： 0 ≤ f ≤ 255
a ∈ [0,1]

操作： FFh → f

受影响的状态位： 无

机器码：

0110	100a	ffff	ffff

说明： 将指定寄存器的内容置为 FFh。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 "立即数变址寻址模式中面向字节和位的指令"**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写 寄存器 f

示例：　SETF　REG,1

执行指令前
REG = 5Ah
执行指令后
REG = FFh

SLEEP 进入休眠模式

语法： SLEEP

操作数： 无

操作： 00h → WDT，
0 → WDT 后分频器，
1 → $\overline{TO}$，
0 → $\overline{PD}$

受影响的状态位： $\overline{TO}$ 和 $\overline{PD}$

机器码：

0000	0000	0000	0011

说明： 掉电状态位（$\overline{PD}$）清零。超时状态位（$\overline{TO}$）置 1。看门狗定时器及其后分频器清零。
振荡器停振，处理器进入休眠模式。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	进入休眠模式

示例： SLEEP

执行指令前
$\overline{TO}$ = ?
$\overline{PD}$ = ?

执行指令后
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† 如果由 WDT 引起唤醒，则此位将被清零。

SUBFWB W 减去 f（带借位）

语法： SUBFWB f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (W) – (f) – ($\overline{C}$) → dest

受影响的状态位： N、OV、C、DC 和 Z

机器码：

0101	01da	ffff	ffff

说明： 将 W 的内容减去 f 寄存器的内容和进位（借位）（通过二进制补码方法进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。
如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节“立即数变址寻址模式中面向字节和位的指令”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1： SUBFWB REG, 1, 0

执行指令前
REG = 3
W = 2
C = 1

执行指令后
REG = FF
W = 2
C = 0
Z = 0
N = 1 ；结果为负

例 2： SUBFWB REG, 0, 0

执行指令前
REG = 2
W = 5
C = 1

执行指令后
REG = 2
W = 3
C = 1
Z = 0
N = 0 ；结果为正

例 3： SUBFWB REG, 1, 0

执行指令前
REG = 1
W = 2
C = 0

执行指令后
REG = 0
W = 2
C = 1
Z = 1 ；结果为零
N = 0

SUBLW 立即数减去 W 的内容

语法：	SUBLW k
操作数：	0 ≤ k ≤ 255
操作：	k – (W) → W
受影响的状态位：	N、 OV、 C、 DC 和 Z
机器码：	0000 | 1000 | kkkk | kkkk
说明：	用 8 位立即数 k 减去 W 的内容。结果保存在 W 寄存器中。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入寄存器 W

例 1： SUBLW 02h

执行指令前
W = 01h
C = ?

执行指令后
W = 01h
C = 1 ；结果为正
Z = 0
N = 0

例 2： SUBLW 02h

执行指令前
W = 02h
C = ?

执行指令后
W = 00h
C = 1 ；结果为零
Z = 1
N = 0

例 3： SUBLW 02h

执行指令前
W = 03h
C = ?

执行指令后
W ＝ FFh ； （2 进制补码）
C = 0 ； 结果为负
Z = 0
N = 1

SUBWF **f 减去 W**

语法：	SUBWF f {,d {,a}}
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f) – (W) → dest
受影响的状态位：	N、 OV、 C、 DC 和 Z
机器码：	0101 | 11da | ffff | ffff
说明：	用寄存器 f 中的内容减去 W 寄存器的内容 （通过二进制补码方式进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f （默认）。 如果 a 为 0，选择快速操作存储区。 如果 a 为 1，使用 BSR 选择 GPR 存储区 （默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

例 1： SUBWF REG, 1, 0

执行指令前
REG = 3
W = 2
C = ?

执行指令后
REG = 1
W = 2
C = 1 ；结果为正
Z = 0
N = 0

例 2： SUBWF REG, 0, 0

执行指令前
REG = 2
W = 2
C = ?

执行指令后
REG = 2
W = 0
C = 1 ；结果为零
Z = 1
N = 0

例 3： SUBWF REG, 1, 0

执行指令前
REG = 1
W = 2
C = ?

执行指令后
REG = FFh ； （二进制补码）
W = 2
C = 0 ； 结果为负
Z = 0
N = 1

SUBWFB f 减去 W （带借位）

语法： SUBWFB f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) - (W) - ($\overline{C}$) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0101	10da	ffff	ffff

说明： 用 f 寄存器的内容减去 W 的内容和进位（借位）标志位 （通过二进制补码方式进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1： SUBWFB REG, 1, 0

执行指令前
REG = 19h （0001 1001）
W = 0Dh （0000 1101）
C = 1

执行指令后
REG = 0Ch （0000 1011）
W = 0Dh （0000 1101）
C = 1
Z = 0
N = 0 ；结果为正

例 2： SUBWFB REG, 0, 0

执行指令前
REG = 1Bh （0001 1011）
W = 1Ah （0001 1010）
C = 0

执行指令后
REG = 1Bh （0001 1011）
W = 00h
C = 1
Z = 1 ；结果为零
N = 0

例 3： SUBWFB REG, 1, 0

执行指令前
REG = 03h （0000 0011）
W = 0Eh （0000 1101）
C = 1

执行指令后
REG = F5h （1111 0100）
；[二进制补码]
W = 0Eh （0000 1101）
C = 0
Z = 0
N = 1 ；结果为负

SWAPF 将 f 的高半字节和低半字节交换

语法： SWAPF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

受影响的状态位： 无

机器码：

0011	10da	ffff	ffff

说明： 将寄存器 f 的高半字节和低半字节互相交换。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： SWAPF REG, 1, 0

执行指令前
REG = 53h

执行指令后
REG = 35h

TBLRD 表读

语法： TBLRD (*; *+; *-; +*)

操作数： 无

操作：
如果执行 TBLRD *，
(程序存储器 (TBLPTR)) → TABLAT ；
TBLPTR →不改变；
如果执行 TBLRD *+，
(程序存储器 (TBLPTR)) → TABLAT ；
(TBLPTR) + 1 → TBLPTR ；
如果执行 TBLRD *-，
(程序存储器 (TBLPTR)) → TABLAT ；
(TBLPTR) –1 → TBLPTR ；
如果执行 TBLRD +*，
(TBLPTR) + 1 → TBLPTR ；
(程序存储器 (TBLPTR)) → TABLAT

受影响的状态位：无

机器码：

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

说明：
该指令用于读取程序存储器 （P.M.）的内容。使用表指针 （TBLPTR）对程序存储器进行寻址。
TBLPTR （一个 21 位指针）指向程序存储器中的每个字节。 TBLPTR 的寻址范围为 2 MB。

TBLPTR[0] = 0： 程序存储器字的低有效字节
TBLPTR[0] = 1： 程序存储器字的高有效字节

TBLRD 指令可用如下方法修改 TBLPTR 的值：
- 不变
- 后加
- 后减
- 预加

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作
空操作	空操作 （读程序存储器）	空操作	空操作 （写 TABLAT）

TBLRD 表读 （续）

例 1： TBLRD *+ ;

执行指令前
TABLAT = 55h
TBLPTR = 00A356h
存储单元 （00A356h） = 34h
执行指令后
TABLAT = 34h
TBLPTR = 00A357h

例 2： TBLRD +* ;

执行指令前
TABLAT = AAh
TBLPTR = 01A357h
存储单元 （01A357h） = 12h
存储单元 （01A358h） = 34h
执行指令后
TABLAT = 34h
TBLPTR = 01A358h

TBLWT 表写

语法： TBLWT (*; *+; *-; +*)

操作数： 无

操作：
如果执行 TBLWT*，
(TABLAT) → 保持寄存器；
TBLPTR → 不改变；
如果执行 TBLWT*+，
(TABLAT) → 保持寄存器；
(TBLPTR) + 1 → TBLPTR ；
如果执行 TBLWT*-，
(TABLAT) → 保持寄存器；
(TBLPTR) –1 → TBLPTR ；
如果执行 TBLWT+*，
(TBLPTR) + 1 → TBLPTR ；
(TABLAT) → 保持寄存器；

受影响的状态位：无

机器码：

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

说明：
此指令使用 TBLPTR 的低 3 位来确定要将 TABLAT 中的内容写入 8 个保持寄存器中的哪一个。该保持寄存器用于对程序存储器（P.M.）的内容编程。（关于对闪存存储器编程的更多详情，请参见**第 6.0 节 “闪存程序存储器”**。）

TBLPTR （一个 21 位指针）指向程序存储器中的每个字节。 TBLPTR 的寻址范围为 2 MB。TBLPTR 的 LSb 选择要访问的程序存储器单元。

TBLPTR[0] = 0： 程序存储器字的低有效字节
TBLPTR[0] = 1： 程序存储器字的高有效字节

TBLWT 指令可用如下方法修改 TBLPTR 的值：
- 不变
- 后加
- 后减
- 预加

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作
空操作	空操作（读 TABLAT）	空操作	空操作（写保持寄存器）

TBLWT 表写 （续）

例 1： `TBLWT *+;`

执行指令前
- TABLAT = 55h
- TBLPTR = 00A356h
- 保持寄存器 (00A356h) = FFh

执行指令后 （表写操作完成）
- TABLAT = 55h
- TBLPTR = 00A357h
- 保持寄存器 (00A356h) = 55h

例 2： `TBLWT +*;`

执行指令前
- TABLAT = 34h
- TBLPTR = 01389Ah
- 保持寄存器 (01389Ah) = FFh
- 保持寄存器 (01389Bh) = FFh

执行指令后 （表写操作完成）
- TABLAT = 34h
- TBLPTR = 01389Bh
- 保持寄存器 (01389Ah) = FFh
- 保持寄存器 (01389Bh) = 34h

TSTFSZ 测试 f，为 0 则跳过

语法：	TSTFSZ f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	f 为 0 则跳过
受影响的状态位：	无
机器码：	0110 011a ffff ffff
说明：	如果 f = 0，丢弃已经取到的指令，转而执行一条 NOP 指令，使这条指令成为双周期指令。 如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。 如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节 “立即数变址寻址模式中面向字节和位的指令”**。
指令字数：	1
指令周期数：	1（2） **注：** 如果跳过的指令后面跟有双字指令，则执行该指令需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过的指令后面跟有双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```

执行指令前
PC = 地址 （HERE）
执行指令后
如果 CNT = 00h，
PC = 地址 （ZERO）
如果 CNT ≠ 00h，
PC = 地址 （NZERO）

XORLW 立即数与 W 作逻辑异或运算

语法：	XORLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .XOR. k → W
受影响的状态位：	N 和 Z
机器码：	0000 1010 kkkk kkkk
说明：	将 W 的内容与 8 位立即数 k 进行逻辑异或运算。结果保存在 W 寄存器中。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入寄存器 W

示例：

```
XORLW   0AFh
```

执行指令前
W = B5h
执行指令后
W = 1Ah

XORWF　　W 与 f 作逻辑异或运算

语法：　XORWF　f {,d {,a}}

操作数：　$0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

操作：　(W) .XOR.(f) → dest

受影响的状态位：　N 和 Z

机器码：

0001	10da	ffff	ffff

说明：　将 W 的内容与寄存器 f 的内容进行逻辑异或运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区（默认）。
如果 a 为 0 且使能了扩展的指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 21.2.3 节“立即数变址寻址模式中面向字节和位的指令”**。

指令字数：　1

指令周期数：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例：　XORWF　REG, 1, 0

执行指令前
REG = AFh
W = B5h

执行指令后
REG = 1Ah
W = B5h

21.2 扩展的指令集

除了 PIC18 指令集的 75 条标准指令外， PIC18F1230/1330 器件还提供了针对 CPU 内核功能的可选扩展指令。这些新增的功能包括 8 条额外的指令，它们可以实现间接和变址寻址操作；以及为许多标准PIC18指令实现了立即数变址寻址模式。

这些扩展指令集的额外功能在默认情况下是禁止的。用户必须通过将 XINST 配置位置 1，才能使能它们。

扩展指令集中的指令（除 CALLW、 MOVSF 和 MOVSS）可以全部被归为立即数操作类指令，它们既可以控制文件选择寄存器，也可以使用这些寄存器进行变址寻址。其中的两条指令 ADDFSR 和 SUBFSR，可以直接对 FSR2 进行操作；而 ADDULNK 和 SUBULNK 指令允许在执行后自动返回。

这些扩展的指令专门用于优化用高级语言，特别是 C 语言编写的重入程序代码（也就是递归调用或使用软件堆栈的代码）。此外，它们使用户能更有效地用高级语言对数据结构执行特定的操作。这些操作包括：

- 在进入和退出子程序时对软件堆栈空间进行动态分配和释放
- 功能指针调用
- 对软件堆栈指针进行控制
- 对软件堆栈中的变量进行控制

表 21-3 提供了扩展指令集中的指令汇总。**第 21.2.2 节 "扩展的指令集"** 对这些指令进行了详细说明。表 21-1（第 208 页）提供了标准和扩展的 PIC18 指令集的操作码字段说明。

注：	扩展的指令集和立即数变址寻址模式是专为优化用 C 语言编写的应用程序而设计的，用户可能不会在汇编器中直接使用这些指令。对于那些需要查看编译器生成代码的用户，这些命令的语法可作为参考。

21.2.1 扩展指令的语法

大部分扩展指令都使用变址参数，同时使用一个文件选择寄存器和某一偏移量来指定源寄存器或目标寄存器。当指令的参数作为变址寻址的一部分时，会用方括号（"[ ]"）把它括起来。这时表示此参数用作变址地址或偏移量。如果 MPASM™ 汇编器发现一个变址地址或偏移量没有被括起来，它就会给出错误警告。

当使能扩展的指令集时，括号也用于表示面向字节和面向位的指令中的变址参数。这是对指令语法的额外更改。欲知更多信息，请参见**第 21.2.3.1 节 "标准 PIC18 命令的扩展指令语法"**。

注：	以前，在 PIC18 和早期的指令集中使用方括号来表示可选参数。在此文本和以后的文本中，可选参数将用大扩号（"{ }"）表示。

表 21-3： PIC18 指令集的扩展

助记符，操作数		说明	指令周期数	16 位指令字				受影响的状态位
				MSb			LSb	
ADDFSR	f, k	将立即数与 FSR 相加	1	1110	1000	ffkk	kkkk	无
ADDULNK	k	将立即数与 FSR2 相加并返回	2	1110	1000	11kk	kkkk	无
CALLW		使用 WREG 调用子程序	2	0000	0000	0001	0100	无
MOVSF	z_s, f_d	将 z_s（源）装入第一个字 将 f_d（目标）装入第二个字	2	1110 1111	1011 ffff	0zzz ffff	zzzz ffff	无
MOVSS	z_s, z_d	将 z_s（源）装入第一个字 将 z_d（目标）装入第二个字	2	1110 1111	1011 xxxx	1zzz xzzz	zzzz zzzz	无
PUSHL	k	将立即数保存在 FSR2 后，FSR2 减 1	1	1110	1010	kkkk	kkkk	无
SUBFSR	f, k	FSR 减去立即数	1	1110	1001	ffkk	kkkk	无
SUBULNK	k	FSR2 减去立即数并返回	2	1110	1001	11kk	kkkk	无

21.2.2 扩展的指令集

ADDFSR **FSR 的内容与立即数相加**

语法： ADDFSR f, k

操作数： 0 ≤ k ≤ 63
f ∈ [0, 1, 2]

操作： FSR(f) + k → FSR(f)

受影响的状态位： 无

机器码：

1110	1000	ffkk	kkkk

说明： 将由 f 指定的 FSR 的内容加上一个 6 位的立即数 k。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入 FSR

示例： `ADDFSR 2, 23h`

执行指令前
FSR2 = 03FFh
执行指令后
FSR2 = 0422h

ADDULNK **FSR2 的内容与立即数相加并返回**

语法： ADDULNK k

操作数： 0 ≤ k ≤ 63

操作： FSR2 + k → FSR2，
(TOS) → PC

受影响的状态位： 无

机器码：

1110	1000	11kk	kkkk

说明： 将 FSR2 的内容加上一个 6 位的立即数 k。然后通过将 TOS 装入 PC，执行一条 RETURN 指令。
执行该指令需要两个周期；在第二个周期执行一条 NOP 指令。
该指令可以被认为是 ADDFSR 指令的特例，其中 f = 3 （二进制 11）；它仅针对 FSR2 进行操作。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理数据	写入 FSR
空操作	空操作	空操作	空操作

示例： `ADDULNK 23h`

执行指令前
FSR2 = 03FFh
PC = 0100h
执行指令后
FSR2 = 0422h
PC = （TOS）

注：	所有的 PIC18 指令都可能在其指令助记符之前使用可选的标号参数，用于符号寻址。如果使用标号，那么指令语法将变为：{ 标号 } 指令参数。

CALLW 使用 **WREG** 调用子程序

语法： CALLW

操作数： 无

操作： （PC + 2）→ TOS，
(W) → PCL，
(PCLATH) → PCH，
(PCLATU) → PCU

受影响的状态位： 无

机器码：

0000	0000	0001	0100

说明 首先，返回地址 （PC + 2）被压入返回堆栈。接下来，将 W 寄存器的内容写入 PCL， PCL 现有的值被丢弃。然后，PCLATH 和 PCLATU 的内容被分别锁存到 PCH 和 PCU。第二个周期执行一条 NOP 指令，并同时取出下一条指令。
和 CALL 不一样，该指令没有更新 W、STATUS 或 BSR 寄存器的选项。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 WREG	将 PC 压入堆栈	空操作
空操作	空操作	空操作	空操作

示例： HERE CALLW

执行指令前
PC = 地址 （HERE）
PCLATH = 10h
PCLATU = 00h
W = 06h

执行指令后
PC = 001006h
TOS = 地址 （HERE + 2）
PCLATH = 10h
PCLATU = 00h
W = 06h

MOVSF 将变址寻址单元内容送入 **f**

语法： MOVSF [z_s], f_d

操作数： $0 \leqslant z_s \leqslant 127$
$0 \leqslant f_d \leqslant 4095$

操作： $((FSR2) + z_s) \rightarrow f_d$

受影响的状态位： 无

机器码：

第一个字 （源）	1110	1011	0zzz	$zzzz_s$
第二个字 （目标）	1111	ffff	ffff	$ffff_d$

说明： 将源寄存器的内容移入目标寄存器 f_d。通过将第一个字中的 7 位立即数偏移量 z_s 与 FSR2 的值相加，来确定源寄存器的实际地址。第二个字中的 12 位立即数 f_d 指向目标寄存器的地址。两个地址均可以是 4096 字节的数据空间 （000h 到 FFFh）中的任何位置。
MOVSF 指令中的目标寄存器不能是 PCL、 TOSU、 TOSH 或 TOSL。
如果计算得到的源地址指向间接寻址寄存器，将返回 00h。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读源寄存器
译码	空操作 非无效读取	空操作	写目标寄存器 f

示例： MOVSF [05h], REG2

执行指令前
FSR2 = 80h
85h 单元的内容 = 33h
REG2 = 11h

执行指令后
FSR2 = 80h
85h 单元的内容 = 33h
REG2 = 33h

MOVSS 变址寻址传送数据

语法： MOVSS [z_s], [z_d]

操作数： $0 \leqslant z_s \leqslant 127$
$0 \leqslant z_d \leqslant 127$

操作： $((FSR2) + z_s) \rightarrow ((FSR2) + z_d)$

受影响的状态位： 无

机器码：

第一个字 （源）	1110	1011	1zzz	$zzzz_s$
第二个字 （目标）	1111	xxxx	xzzz	$zzzz_d$

说明： 将源寄存器的内容送入目标寄存器。通过将 FSR2 中的值分别加上 7 位立即数偏移量 z_s 和 z_d 来确定源寄存器和目标寄存器的地址。两个寄存器都可以是 4096 字节数据存储空间 （000h 到 FFFh）中的任意单元。
MOVSS 指令中的目标寄存器不能是 PCL、 TOSU、 TOSH 或 TOSL。
如果计算得到的源地址指向间接寻址寄存器，将返回 00h。如果计算得到的目标地址指向间接寻址寄存器，指令将作为一条 NOP 指令执行。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读源寄存器
译码	确定目标地址	确定目标地址	写目标寄存器

示例： `MOVSS [05h], [06h]`

执行指令前
- FSR2 = 80h
- 85h 单元的内容 = 33h
- 86h 单元的内容 = 11h

执行指令后
- FSR2 = 80h
- 85h 单元的内容 = 33h
- 86h 单元的内容 = 33h

PUSHL 将立即数保存在 FSR2，FSR2 减 1

语法： PUSHL k

操作数： $0 \leqslant k \leqslant 255$

操作： k → （FSR2），
FSR2 – 1 → FSR2

受影响的状态位：无

机器码：

1111	1010	kkkk	kkkk

说明： 8 位立即数 k 被写入由 FSR2 指定的数据存储单元。操作完后 FSR2 减 1。
此指令允许用户将值压入软件堆栈。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标寄存器

示例： `PUSHL 08h`

执行指令前
- FSR2H:FSR2L = 01ECh
- 存储单元 （01ECh） = 00h

执行指令后
- FSR2H:FSR2L = 01EBh
- 存储单元 （01ECh） = 08h

SUBFSR FSR 减去立即数

语法：	SUBFSR f, k
操作数：	0 ≤ k ≤ 63 f ∈ [0, 1, 2]
操作：	FSR(f – k) → FSR(f)
受影响的状态位：	无
机器码：	`1110` `1001` `ffkk` `kkkk`
说明：	用 f 指定的 FSR 的内容减去 6 位立即数 k。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器

示例： `SUBFSR 2, 23h`

执行指令前
FSR2 = 03FFh

执行指令后
FSR2 = 03DCh

SUBULNK FSR2 减去立即数并返回

语法：	SUBULNK k
操作数：	0 ≤ k ≤ 63
操作：	FSR2 – k → FSR2 (TOS) → PC
受影响的状态位：	无
机器码：	`1110` `1001` `11kk` `kkkk`
说明：	将 FSR2 的内容减去 6 位立即数 k，然后通过将 TOS 装入 PC，执行一条 RETURN 指令。 执行该指令需要两个指令周期，第二个指令周期执行一条 NOP 指令。 该指令可以被认为是 SUBFSR 指令的特例，其中 f = 3 （二进制 11）；它仅针对 FSR2 进行操作。
指令字数：	1
指令周期数：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	写入 目标寄存器
空操作	空操作	空操作	空操作

示例： `SUBULNK 23h`

执行指令前
FSR2 = 03FFh
PC = 0100h

执行指令后
FSR2 = 03DCh
PC = (TOS)

21.2.3 立即数变址寻址模式中面向字节和位的指令

注：	使能 PIC18 扩展指令集可能导致常规应用程序运行不正常或完全失败。

一旦使能扩展的指令集，除了可以使用 8 条新命令之外，还可以使用立即数变址寻址模式（**第 5.5.1 节“用立即数偏移量进行变址寻址”**）。这将导致许多标准PIC18 指令的地址解析方法有很大变化。

当禁用扩展的指令集时，被嵌入在操作码中的地址被视作立即数存储单元地址：可以是快速操作存储区中的单元 （a = 0），或由 BSR 指定的 GPR 存储区中的单元（a = 1）。当使能扩展的指令集且 a = 0 时，地址为 5Fh或以下的文件寄存器参数被解析为 FSR2 中的指针值的偏移量，而不是一个立即数地址。对于实际应用来说，这意味着所有使用快速操作 RAM 位作为参数的指令，即所有面向字节或位的指令，或者几乎半数的 PIC18 内核指令——在使能了扩展的指令集时操作都会有所不同。

当 FSR2 的内容为 00h 时，快速操作 RAM 的边界会被重新映射到它们的原始值。这对于编写向下兼容的代码很有用处。如果使用此技术，有必要在 C 程序调用汇编子程序时保存 FSR2 的值并在返回时将它恢复，这样做的目的是保护堆栈指针。用户还必须记住扩展指令集的语法要求 （见**第 21.2.3.1 节 “标准 PIC18 命令的扩展指令语法”**）。

虽然立即数变址寻址模式对于动态堆栈和指针控制很有用处，但是如果不小心误用了寄存器，也会非常麻烦。已经习惯使用 PIC18 编程的用户必须记住，在使能了扩展的指令集时，地址小于或等于 5Fh 的寄存器用于立即数变址寻址模式。

下面是在立即数变址寻址模式中，一些面向字节和位的指令的示例，通过示例可以看出指令如何受到影响。示例中的操作数条件适用于所有这一类的指令。

21.2.3.1 标准 PIC18 命令的扩展指令语法

当使能了扩展的指令集时，立即数偏移量 k 被用来替换标准的面向字节和位的命令中的文件寄存器参数 f。如前所述，只有在 f 小于或等于 5Fh 时才会发生这种情况。当使用偏移量时，该偏移量必须用方括号 “[ ]” 标出。因为在扩展的指令集中，括号中的数值被解析为变址地址或偏移量。省略括号，或在括号内使用大于 5Fh 的值会在 MPASM 汇编器中产生错误。

如果变址参数已被加上了括号，那么就不再需要指定快速操作 RAM 参数；此参数被假定为 0。这与标准操作（禁止扩展的指令集时）正好相反。在变址寻址模式中，声明快速操作 RAM 位也将在 MPASM 汇编器中产生错误。

目标参数 d 的操作和以前一样。

在 MPASM 汇编器的最新版本中，必须明确调用对扩展的指令集的语言支持。可以通过命令行选项 /y 或在源代码中加入 PE 伪指令进行调用。

21.2.4 使能扩展指令集时的注意事项

需要注意的是并非所有用户都有必要使用扩展的指令集，尤其是那些不使用软件堆栈的用户。

此外，立即数变址寻址模式可能会给写入 PIC18 汇编器的常规应用程序带来问题。这是因为常规的指令会尝试寻址快速操作存储区中地址低于 5Fh 的寄存器。当使能了扩展的指令集时，这些地址被解析为相对于 FSR2 的立即数偏移量，所以应用程序会读或写错误的地址。

将应用程序移植到 PIC18F1230/1330 器件时，代码类型是非常重要的。在使用扩展的指令集时，用 C 语言编写的代码较长的重入应用程序会运行的很好，而大量使用快速操作存储区的常规应用程序不会获得任何益处。

ADDWF 将 W 与变址寻址单元的内容相加（立即数变址寻址模式）

语法：	ADDWF [k] {,d}
操作数：	0 ≤ k ≤ 95 d ∈ [0,1]
操作：	(W) + ((FSR2) + k) → dest
受影响的状态位：	N、OV、C、DC 和 Z
机器码：	0010 01d0 kkkk kkkk
说明：	将 W 的内容与由 FSR2 加上偏移量 k 指定的寄存器的内容相加。 如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f（默认）。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标寄存器

示例： ADDWF [OFST] ,0

执行指令前
- W = 17h
- OFST = 2Ch
- FSR2 = 0A00h
- 0A2Ch 单元的内容 = 20h

执行指令后
- W = 37h
- 0A2Ch 单元的内容 = 20h

BSF 将变址寻址单元相应位置 1（立即数变址寻址模式）

语法：	BSF [k], b
操作数：	0 ≤ f ≤ 95 0 ≤ b ≤ 7
操作：	1 → ((FSR2) + k)<b>
受影响的状态位：	无
机器码：	1000 bbb0 kkkk kkkk
说明：	将由 FSR2 加上偏移量 k 指定的寄存器中的位 b 置 1。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： BSF [FLAG_OFST], 7

执行指令前
- FLAG_OFST = 0Ah
- FSR2 = 0A00h
- 0A0Ah 单元的内容 = 55h

执行指令后
- 0A0Ah 单元的内容 = D5h

SETF 将变址寻址单元置全 1（立即数变址寻址模式）

语法：	SETF [k]
操作数：	0 ≤ k ≤ 95
操作：	FFh → ((FSR2) + k)
受影响的状态位：	无
机器码：	0110 1000 kkkk kkkk
说明：	将由 FSR2 加上偏移量 k 指定的寄存器的内容置为 FFh。
指令字数：	1
指令周期数：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写寄存器

示例： SETF [OFST]

执行指令前
- OFST = 2Ch
- FSR2 = 0A00h
- 0A2Ch 单元的内容 = 00h

执行指令后
- 0A2Ch 单元的内容 = FFh

21.2.5 使用 MICROCHIP MPLAB® IDE 工具的注意事项

最新版本的Microchip软件工具，完全支持PIC18F1230/1330 系列器件的扩展指令集。包括 MPLAB C18 C 编译器、 MPASM 汇编语言和 MPLAB 集成开发环境（Integrated Development Environment ， IDE）。

在选择了用于软件开发的目标器件后， MPLAB IDE 将对该器件的默认配置进行自动设置。XINST 配置位的默认设置是 0，即禁用扩展的指令集和立即数变址寻址模式。在编程过程中必须将 XINST 位置 1 才能正确使用扩展指令集开发应用程序。

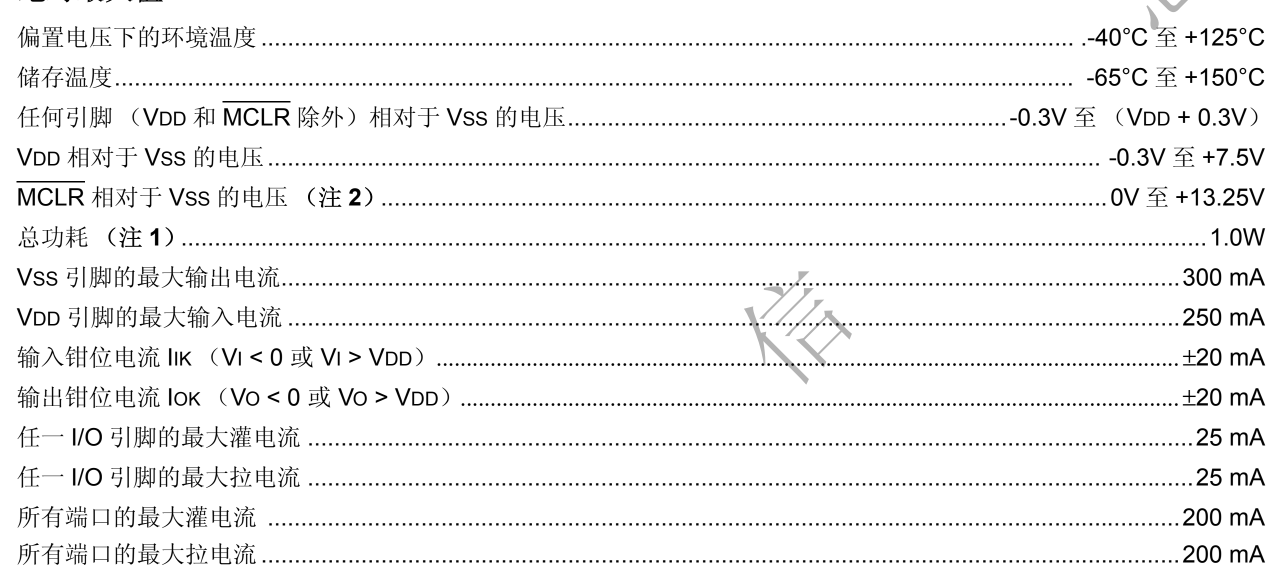
要使用扩展的指令集开发软件，用户必须设置他们的语言工具以实现对扩展指令和变址寻址模式的支持。根据所使用的环境，可以通过以下几种方法：

- 开发环境中的菜单选项或对话框，允许用户配置项目的语言工具及其设置
- 命令行选项
- 源代码中的伪指令

这些选项在不同的编译器、汇编器和开发环境中将有所不同。建议用户在其开发系统所附带的文档中查询相应的信息。

22.0 电气规范

绝对最大值 (†)



偏置电压下的环境温度 .... -40°C 至 +125°C
储存温度 .... -65°C 至 +150°C
任何引脚 （V_{DD} 和 $\overline{MCLR}$ 除外）相对于 V_{SS} 的电压 .... -0.3V 至 （V_{DD} + 0.3V）
V_{DD} 相对于 V_{SS} 的电压 .... -0.3V 至 +7.5V
$\overline{MCLR}$ 相对于 V_{SS} 的电压 （注 **2**）.. 0V 至 +13.25V
总功耗 （注 **1**）.. 1.0W
V_{SS} 引脚的最大输出电流 .... 300 mA
V_{DD} 引脚的最大输入电流 .... 250 mA
输入钳位电流 I_{IK} （V_I < 0 或 V_I > V_{DD}）.. ±20 mA
输出钳位电流 I_{OK} （V_O < 0 或 V_O > V_{DD}）.. ±20 mA
任一 I/O 引脚的最大灌电流 .... 25 mA
任一 I/O 引脚的最大拉电流 .... 25 mA
所有端口的最大灌电流 .... 200 mA
所有端口的最大拉电流 .... 200 mA

注 **1**：功耗按如下公式计算：
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2：如果 $\overline{MCLR}$/V_{PP}/RA5/$\overline{FLTA}$ 引脚上的尖峰电压低于 V_{SS}，感应电流大于 80mA，可能会引起器件锁死。因此当 $\overline{MCLR}$/V_{PP}/RA5/$\overline{FLTA}$ 引脚驱动为低电平时，应该串联一个 50 － 100Ω 的电阻，而不是直接把该引脚连接到 V_{SS}。

† 注意：如果器件工作条件超过上述 “绝对最大值”，可能会对器件造成永久性损坏。上述值仅为运行条件极大值，我们不建议器件在该规范规定的范围以外运行。器件长时间工作在最大值条件下，其稳定性会受到影响。



图 22-1： PIC18F1230/1330 电压—频率关系图 （工业级）

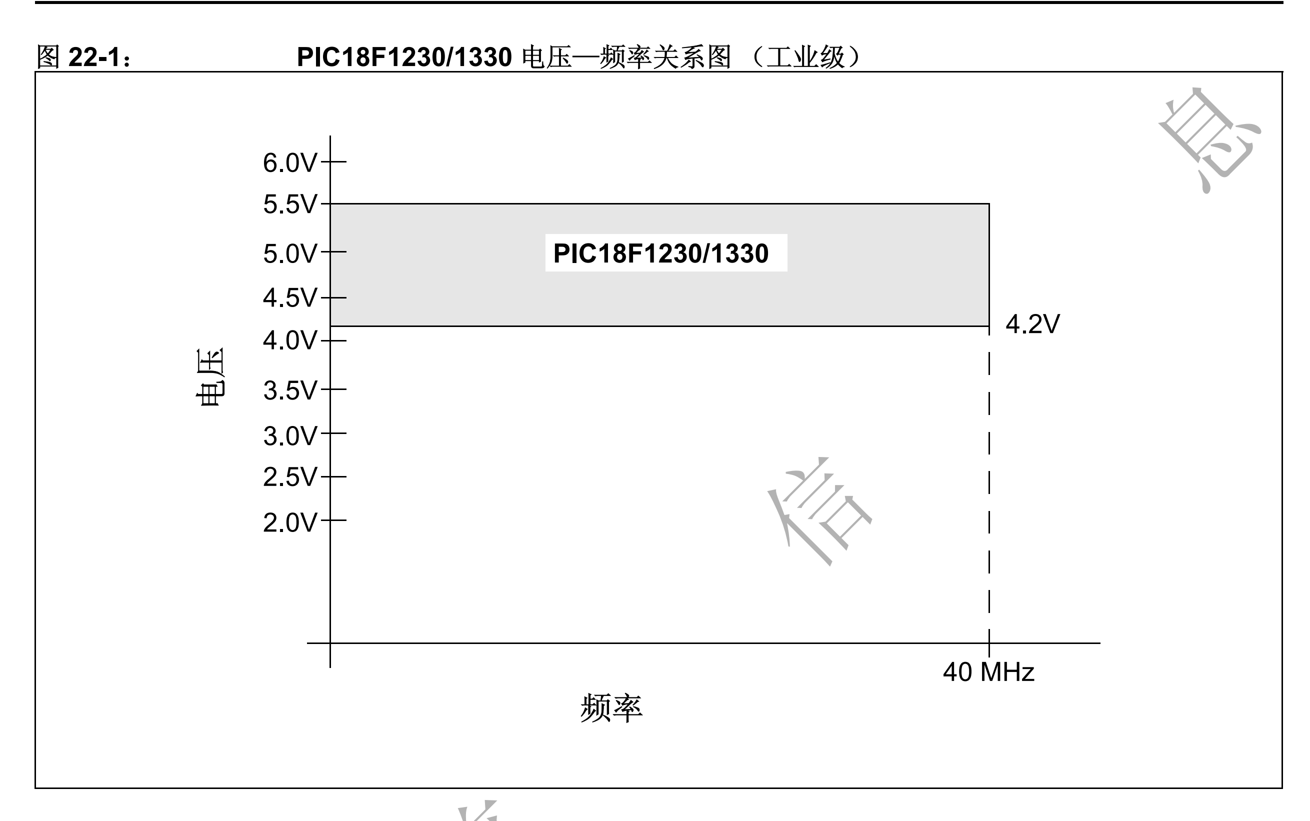


图 22-2： PIC18F1230/1330 电压—频率关系图 （扩展级）

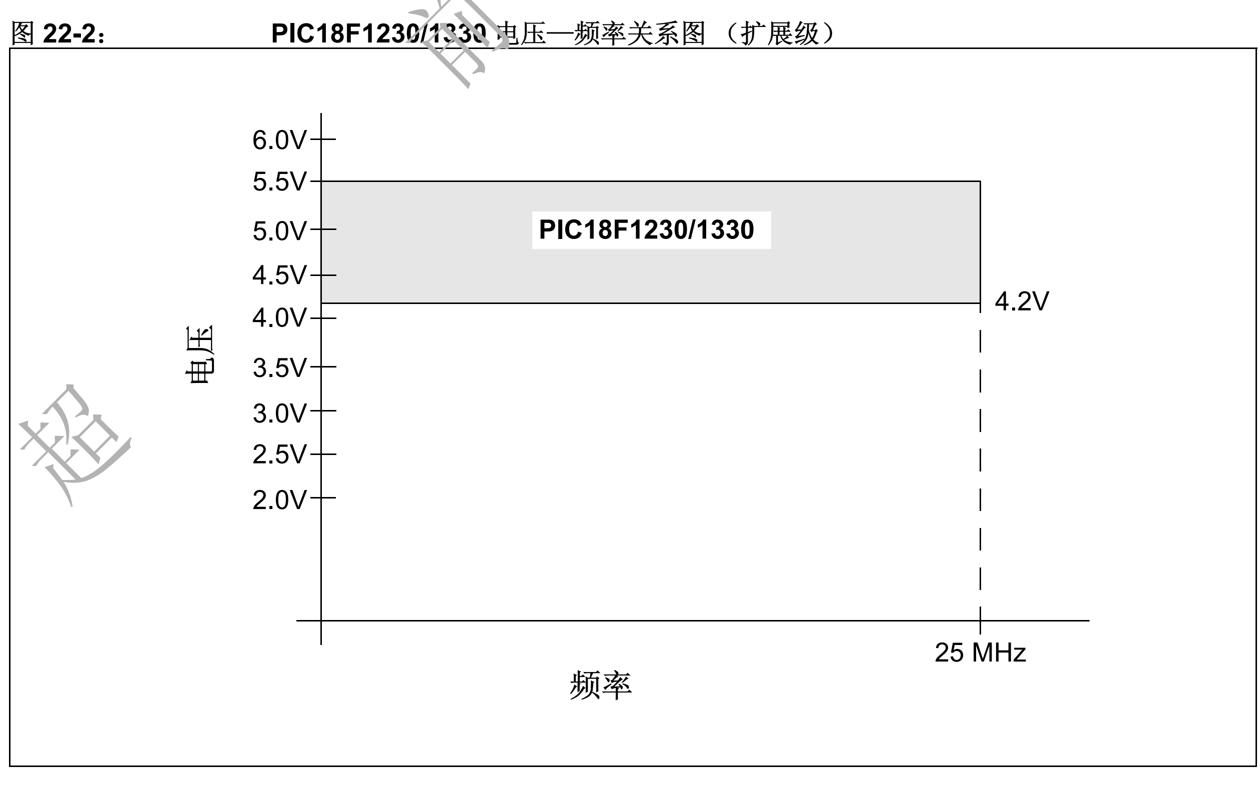


图 **22-3：** **PIC18LF1230/1330** 电压—频率关系图 （工业级）

6.0V
5.5V
5.0V
4.5V
4.0V
3.5V
3.0V
2.5V
2.0V

电压

PIC18LF1230/1330

4.2V

4 MHz
40 MHz

频率

F_{MAX} = (16.36 MHz/V) ($V_{DDAPPMIN}$ –2.0V) + 4 MHz

注：$V_{DDAPPMIN}$ 是 PIC® 器件在应用中的最小电压。

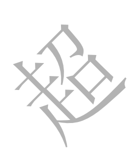


22.1 直流规范： 供电电压
PIC18F1230/1330 （工业级）
PIC18LF1230/1330 （工业级）

PIC18LF1230/1330 （工业级）			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
PIC18F1230/1330 （工业级和扩展级）			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	符号	特性	最小值	典型值	最大值	单位	条件
D001	VDD	供电电压					
		PIC18LF1230/1330	2.0	—	5.5	V	HS、 XT、 RC 和 LP 振荡器模式
		PIC18F1230/1330	4.2	—	5.5	V	
D002	VDR	**RAM 数据保存电压** (1)	1.5	—	—	V	
D003	VPOR	**VDD 启动电压** （确保内部上电复位信号）	—	—	0.7	V	详情请参见上电复位章节
D004	SVDD	**VDD 上升率** （确保内部上电复位信号）	0.05	—	—	V/ms	详情请参见上电复位章节
	VBOR	欠压复位电压					
D005		PIC18LF1230/1330					
		BORV1:BORV0 = 11	2.00	2.05	2.16	V	
		BORV1:BORV0 = 10	2.65	2.79	2.93	V	
D005		所有器件					
		BORV1:BORV0 = 01	4.11	4.33	4.55	V	
		BORV1:BORV0 = 00	4.36	4.59	4.82	V	

图注： 阴影行是为了增强表的可读性。

注 1： 该电压是在休眠模式或器件复位状态下，在不丢失 RAM 数据的前提下的最小 VDD。





22.2 直流规范： 掉电电流和供电电流
PIC18F1230/1330 （工业级）
PIC18LF1230/1330 （工业级）

PIC18LF1230/1330 （工业级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ T_A ≤ +85°C （工业级）
PIC18F1230/1330 （工业级和扩展级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） -40°C ≤ T_A ≤ +125°C （扩展级）

参数编号	器件	典型值	最大值	单位	条件	
	掉电电流 （I_{PD}）[1]					
	PIC18LF1230/1330	100	TBD	nA	-40°C	V_{DD} = 2.0V （休眠模式）
		0.1	TBD	μA	+25°C	
		0.2	TBD	μA	+85°C	
	PIC18LF1230/1330	0.1	TBD	μA	-40°C	V_{DD} = 3.0V （休眠模式）
		0.1	TBD	μA	+25°C	
		0.3	TBD	μA	+85°C	
	所有器件	0.1	TBD	μA	-40°C	V_{DD} = 5.0V （休眠模式）
		0.1	TBD	μA	+25°C	
		0.4	TBD	μA	+85°C	
	仅扩展器件	10	TBD	μA	+125°C	

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 V_{DD} 或 V_{SS}，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 I_{DD} 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 V_{DD} 或 V_{SS}；
MCLR = V_{DD}；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。



22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）		**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ T_A ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级和扩展级）		**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） -40°C ≤ T_A ≤ +125°C （扩展级）					
参数编号	**器件**	**典型值**	**最大值**	**单位**	**条件**		
	供电电流 （**I_{DD}**）[2]						
	PIC18LF1230/1330	15	TBD	μA	-40°C	V_{DD} = 2.0V	F_{OSC} = 31 kHz （**RC_RUN** 模式， INTRC 时钟源）
		15	TBD	μA	+25°C		
		15	TBD	μA	+85°C		
	PIC18LF1230/1330	40	TBD	μA	-40°C	V_{DD} = 3.0V	
		35	TBD	μA	+25°C		
		30	TBD	μA	+85°C		
	所有器件	105	TBD	μA	-40°C	V_{DD} = 5.0V	
		90	TBD	μA	+25°C		
		80	TBD	μA	+85°C		
	仅扩展型器件	80	TBD	μA	+125°C		
	PIC18LF1230/1330	0.32	TBD	mA	-40°C	V_{DD} = 2.0V	F_{OSC} = 1 MHz （**RC_RUN** 模式， INTOSC 时钟源）
		0.33	TBD	mA	+25°C		
		0.33	TBD	mA	+85°C		
	PIC18LF1230/1330	0.6	TBD	mA	-40°C	V_{DD} = 3.0V	
		0.55	TBD	mA	+25°C		
		0.6	TBD	mA	+85°C		
	所有器件	1.1	TBD	mA	-40°C	V_{DD} = 5.0V	
		1.1	TBD	mA	+25°C		
		1.0	TBD	mA	+85°C		
	仅扩展型器件	1	TBD	mA	+125°C		

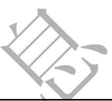
图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 V_{DD} 或 V_{SS}，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 I_{DD} 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 V_{DD} 或 V_{SS}；
MCLR = V_{DD}；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。



22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级）
PIC18F1230/1330 （工业级和扩展级）	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）

参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD） (2)						
	PIC18LF1230/1330	0.8	TBD	μA	-40°C	VDD = 2.0V	FOSC = 4 MHz（RC_RUN 模式，INTOSC 时钟源）
		0.8	TBD	μA	+25°C		
		0.8	TBD	μA	+85°C		
	PIC18LF1230/1330	1.3	TBD	mA	-40°C	VDD = 3.0V	
		1.3	TBD	mA	+25°C		
		1.3	TBD	mA	+85°C		
	所有器件	2.5	TBD	mA	-40°C	VDD = 5.0V	
		2.5	TBD	mA	+25°C		
		2.5	TBD	mA	+85°C		
	仅扩展型器件	2.5	TBD	mA	+125°C		
	PIC18LF1230/1330	2.9	TBD	μA	-40°C	VDD = 2.0V	FOSC = 31 kHz（RC_IDLE 模式，INTRC 时钟源）
		3.1	TBD	μA	+25°C		
		3.6	TBD	μA	+85°C		
	PIC18LF1230/1330	4.5	TBD	μA	-40°C	VDD = 3.0V	
		4.8	TBD	μA	+25°C		
		5.8	TBD	μA	+85°C		
	所有器件	9.2	TBD	μA	-40°C	VDD = 5.0V	
		9.8	TBD	μA	+25°C		
		11.4	TBD	μA	+85°C		
	仅扩展型器件	21	TBD	μA	+125°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS ；
MCLR = VDD ；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。

22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ T_A ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级和扩展级）		**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） -40°C ≤ T_A ≤ +125°C （扩展级）					
参数编号	**器件**	**典型值**	**最大值**	**单位**	**条件**		
	供电电流 （I_{DD}）(2)						
	PIC18LF1230/1330	165	TBD	µA	-40°C	V_{DD} = 2.0V	F_{OSC} = 1 MHz （**RC_IDLE** 模式， INTOSC 时钟源）
		175	TBD	µA	+25°C		
		190	TBD	µA	+85°C		
	PIC18LF1230/1330	250	TBD	µA	-40°C	V_{DD} = 3.0V	
		270	TBD	µA	+25°C		
		290	TBD	µA	+85°C		
	所有器件	500	TBD	mA	-40°C	V_{DD} = 5.0V	
		520	TBD	mA	+25°C		
		550	TBD	mA	+85°C		
	仅扩展型器件	0.6	TBD	mA	+125°C		
	PIC18LF1230/1330	340	TBD	µA	-40°C	V_{DD} = 2.0V	F_{OSC} = 4 MHz （**RC_IDLE** 模式， INTOSC 时钟源）
		350	TBD	µA	+25°C		
		360	TBD	µA	+85°C		
	PIC18LF1230/1330	520	TBD	µA	-40°C	V_{DD} = 3.0V	
		540	TBD	µA	+25°C		
		580	TBD	µA	+85°C		
	所有器件	1.0	TBD	mA	-40°C	V_{DD} = 5.0V	
		1.1	TBD	mA	+25°C		
		1.1	TBD	mA	+85°C		
	仅扩展型器件	1.1	TBD	mA	+125°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 V_{DD} 或 V_{SS}，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 I_{DD} 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 V_{DD} 或 V_{SS} ；
MCLR = V_{DD} ；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。



22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级和扩展级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD） (2)						
	PIC18LF1230/1330	250	TBD	μA	-40°C	VDD = 2.0V	FOSC = 1 MHz （PRI_RUN， EC 振荡器）
		260	TBD	μA	+25°C		
		250	TBD	μA	+85°C		
	PIC18LF1230/1330	550	TBD	μA	-40°C	VDD = 3.0V	
		480	TBD	μA	+25°C		
		460	TBD	μA	+85°C		
	所有器件	1.2	TBD	mA	-40°C	VDD = 5.0V	
		1.1	TBD	mA	+25°C		
		1.0	TBD	mA	+85°C		
	仅扩展型器件	1.0	TBD	mA	+125°C		
	PIC18LF1230/1330	0.72	TBD	mA	-40°C	VDD = 2.0V	FOSC = 4 MHz （PRI_RUN， EC 振荡器）
		0.74	TBD	mA	+25°C		
		0.74	TBD	mA	+85°C		
	PIC18LF1230/1330	1.3	TBD	mA	-40°C	VDD = 3.0V	
		1.3	TBD	mA	+25°C		
		1.3	TBD	mA	+85°C		
	所有器件	2.7	TBD	mA	-40°C	VDD = 5.0V	
		2.6	TBD	mA	+25°C		
		2.5	TBD	mA	+85°C		
	仅扩展型器件	2.6	TBD	mA	+125°C		
	仅扩展型器件	8.4	TBD	mA	+125°C	VDD = 4.2V	FOSC = 25 MHz （PRI_RUN， EC 振荡器）
		11	TBD	mA	+125°C	VDD = 5.0V	
	所有器件	15	TBD	mA	-40°C	VDD = 4.2V	FOSC = 40 MHz （PRI_RUN， EC 振荡器）
		16	TBD	mA	+25°C		
		16	TBD	mA	+85°C		
	所有器件	21	TBD	mA	-40°C	VDD = 5.0V	
		21	TBD	mA	+25°C		
		21	TBD	mA	+85°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS ；
MCLR = VDD ；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。

22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ T_A ≤ +85°C （工业级）						
PIC18F1230/1330 （工业级和扩展级）	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） -40°C ≤ T_A ≤ +125°C （扩展级）						
参数编号	**器件**	**典型值**	**最大值**	**单位**	**条件**		
	供电电流 （I_{DD}）[2]						
	所有器件	7.5	TBD	mA	-40°C	V_{DD} = 4.2V	F_{OSC} = 4 MHz，16 MHz 内部（**PRI_RUN HS+PLL**）
		7.4	TBD	mA	+25°C		
		7.3	TBD	mA	+85°C		
	仅扩展型器件	8.0	TBD	mA	+125°C		
	所有器件	10	TBD	mA	-40°C	V_{DD} = 5.0V	F_{OSC} = 4 MHz，16 MHz 内部（**PRI_RUN HS+PLL**）
		10	TBD	mA	+25°C		
		9.7	TBD	mA	+85°C		
	仅扩展型器件	10	TBD	mA	+125°C		
	所有器件	17	TBD	mA	-40°C	V_{DD} = 4.2V	F_{OSC} = 10 MHz，40 MHz 内部（**PRI_RUN HS+PLL**）
		17	TBD	mA	+25°C		
		17	TBD	mA	+85°C		
	所有器件	23	TBD	mA	-40°C	V_{DD} = 5.0V	F_{OSC} = 10 MHz，40 MHz 内部（**PRI_RUN HS+PLL**）
		23	TBD	mA	+25°C		
		23	TBD	mA	+85°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 V_{DD} 或 V_{SS}，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 I_{DD} 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 V_{DD} 或 V_{SS}；
MCLR = V_{DD}；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。





22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级和扩展级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）(2)						
	PIC18LF1230/1330	65	TBD	μA	-40°C	VDD = 2.0V	FOSC = 1 MHz（**PRI_IDLE** 模式，EC 振荡器）
		65	TBD	μA	+25°C		
		70	TBD	μA	+85°C		
	PIC18LF1230/1330	120	TBD	μA	-40°C	VDD = 3.0V	
		120	TBD	μA	+25°C		
		130	TBD	μA	+85°C		
	所有器件	300	TBD	μA	-40°C	VDD = 5.0V	
		240	TBD	μA	+25°C		
		300	TBD	μA	+85°C		
	仅扩展型器件	320	TBD	μA	+125°C		
	PIC18LF1230/1330	260	TBD	μA	-40°C	VDD = 2.0V	FOSC = 4 MHz（**PRI_IDLE** 模式，EC 振荡器）
		255	TBD	μA	+25°C		
		270	TBD	μA	+85°C		
	PIC18LF1230/1330	420	TBD	μA	-40°C	VDD = 3.0V	
		430	TBD	μA	+25°C		
		450	TBD	μA	+85°C		
	所有器件	0.9	TBD	mA	-40°C	VDD = 5.0V	
		0.9	TBD	mA	+25°C		
		0.9	TBD	mA	+85°C		
	仅扩展型器件	1	TBD	mA	+125°C		
	仅扩展型器件	2.8	TBD	mA	+125°C	VDD = 4.2V	FOSC = 25 MHz（**PRI_IDLE** 模式，EC 振荡器）
		4.3	TBD	mA	+125°C	VDD = 5.0V	
	所有器件	6.0	TBD	mA	-40°C	VDD = 4.2V	FOSC = 40 MHz（**PRI_IDLE** 模式，EC 振荡器）
		6.2	TBD	mA	+25°C		
		6.6	TBD	mA	+85°C		
	所有器件	8.1	TBD	mA	-40°C	VDD = 5.0V	
		9.1	TBD	mA	+25°C		
		8.3	TBD	mA	+85°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS ；
MCLR = VDD ；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。

22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级）
PIC18F1230/1330 （工业级和扩展级）	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）

参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD） (2)						
	PIC18LF1230/1330	14	TBD	μA	-40°C	VDD = 2.0V	FOSC = 32 kHz(4) （**SEC_RUN** 模式， Timer1 作为时钟源）
		15	TBD	μA	+25°C		
		16	TBD	μA	+85°C		
	PIC18LF1230/1330	40	TBD	μA	-40°C	VDD = 3.0V	
		35	TBD	μA	+25°C		
		31	TBD	μA	+85°C		
	所有器件	99	TBD	μA	-40°C	VDD = 5.0V	
		81	TBD	μA	+25°C		
		75	TBD	μA	+85°C		
	PIC18LF1230/1330	2.5	TBD	μA	-40°C	VDD = 2.0V	FOSC = 32 kHz(4) （**SEC_IDLE** 模式， Timer1 作为时钟源）
		3.7	TBD	μA	+25°C		
		4.5	TBD	μA	+85°C		
	PIC18LF1230/1330	5.0	TBD	μA	-40°C	VDD = 3.0V	
		5.4	TBD	μA	+25°C		
		6.3	TBD	μA	+85°C		
	所有器件	8.5	TBD	μA	-40°C	VDD = 5.0V	
		9.0	TBD	μA	+25°C		
		10.5	TBD	μA	+85°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 **1**： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS；
MCLR = VDD；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。



22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级） （续）

PIC18LF1230/1330 （工业级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）						
PIC18F1230/1330 （工业级和扩展级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）						
参数编号	器件	典型值	最大值	单位	条件		
	模块差分电流 （ΔIWDT、 ΔIBOR、 ΔILVD、 ΔIOSCB 和 ΔIAD）						
D022 （ΔIWDT）	看门狗定时器	1.3	TBD	μA	-40°C	VDD = 2.0V	
		1.4	TBD	μA	+25°C		
		2.0	TBD	μA	+85°C		
		1.9	TBD	μA	-40°C	VDD = 3.0V	
		2.0	TBD	μA	+25°C		
		2.8	TBD	μA	+85°C		
		4.0	TBD	μA	-40°C	VDD = 5.0V	
		5.5	TBD	μA	+25°C		
		5.6	TBD	μA	+85°C		
		13	TBD	μA	+125°C		
D022A (ΔIBOR）	欠压复位 (4)	35	TBD	μA	-40°C 至 +85°C	VDD = 3.0V	
		40	TBD	μA	-40°C 至 +85°C	VDD = 5.0V	
		55	TBD	μA	-40°C 至 +125°C		
		0	TBD	μA	-40°C 至 +85°C		休眠模式， BOREN1:BOREN0 = 10
		0	TBD	μA	-40°C 至 +125°C		
D022B (ΔILVD）	低电压检测 (4)	22	TBD	μA	-40°C 至 +85°C	VDD = 2.0V	
		25	TBD	μA	-40°C 至 +85°C	VDD = 3.0V	
		29	TBD	μA	-40°C 至 +85°C	VDD = 5.0V	
		30	TBD	μA	-40°C 至 +125°C		
D025 (ΔIOSCB）	**Timer1 振荡器**	2.1	TBD	μA	-40°C	VDD = 2.0V	Timer1 为 32 kHz(3)
		1.8	TBD	μA	+25°C		
		2.1	TBD	μA	+85°C		
		2.2	TBD	μA	-40°C	VDD = 3.0V	Timer1 为 32 kHz(3)
		2.6	TBD	μA	+25°C		
		2.9	TBD	μA	+85°C		
		3.0	TBD	μA	-40°C	VDD = 5.0V	Timer1 为 32 kHz(3)
		3.2	TBD	μA	+25°C		
		3.4	TBD	μA	+85°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

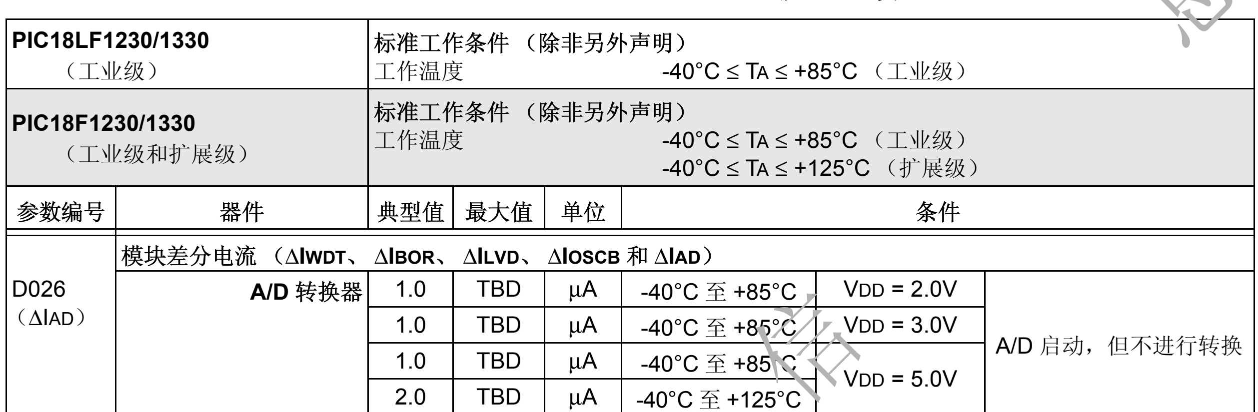
注 **1：** 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS；
MCLR = VDD；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。

22.2 直流规范： 掉电电流和供电电流 PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级）（续）



PIC18LF1230/1330 （工业级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级和扩展级）	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）					
参数编号	**器件**	**典型值**	**最大值**	**单位**	**条件**		
D026 （ΔIAD）	**模块差分电流 （ΔIWDT、 ΔIBOR、 ΔILVD、 ΔIOSCB 和 ΔIAD）**						
	A/D 转换器	1.0	TBD	μA	-40°C 至 +85°C	VDD = 2.0V	A/D 启动，但不进行转换
		1.0	TBD	μA	-40°C 至 +85°C	VDD = 3.0V	
		1.0	TBD	μA	-40°C 至 +85°C	VDD = 5.0V	
		2.0	TBD	μA	-40°C 至 +125°C		

图注： TBD = 待定。阴影行是为了增强表的可读性。

注 1： 休眠模式下的掉电电流不是由振荡器类型决定的。掉电电流是在器件休眠时，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止所有会带来新增电流的功能部件 （比如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要是由工作电压、频率和模式决定的。其他因素，如 I/O 引脚负载和开关频率、振荡器类型和电路、内部代码执行模式和温度也会影响电流消耗。
在正常工作模式下，所有 IDD 测量的测试条件为：
OSC1 = 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD 或 VSS ；
MCLR = VDD ；根据具体应用使能或禁止 WDT。

3： 选择低功耗 Timer1 振荡器。

4： BOR 和 HLVD 使能内部带隙参考源。当这两个模块同时被使能时，电流消耗将少于两个规范值之和。



22.3 直流规范： PIC18F1230/1330 （工业级） PIC18LF1230/1330 （工业级）

直流规范			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）			
参数编号	符号	特性	最小值	最大值	单位	条件
	VIL	输入低电压				
		I/O 端口：				
D030		带 TTL 缓冲器	VSS	0.15 VDD	V	VDD < 4.5V
D030A			—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D031		带施密特触发缓冲器	VSS	0.2 VDD	V	
D032		MCLR	VSS	0.2 VDD	V	
D033		OSC1	VSS	0.3 VDD	V	HS 或 HSPLL 模式
D033A		OSC1	VSS	0.2 VDD	V	RC 或 EC 模式 (1)
D033B		OSC1	VSS	0.3	V	XT 或 LP 模式
D034		T1CKI	VSS	0.3	V	
	VIH	输入高电压				
		I/O 端口：				
D040		带 TTL 缓冲器	0.25 VDD + 0.8V	VDD	V	VDD < 4.5V
D040A			2.0	VDD	V	4.5V ≤ VDD ≤ 5.5V
D041		带施密特触发缓冲器	0.8 VDD	VDD	V	
D042		MCLR	0.8 VDD	VDD	V	
D043		OSC1	0.7 VDD	VDD	V	HS 或 HSPLL 模式
D043A		OSC1	0.8 VDD	VDD	V	EC 模式
D043B		OSC1	0.9 VDD	VDD	V	RC 模式 (1)
D043C		OSC1	1.6	VDD	V	XT 或 LP 模式
D044		T1CKI	1.6	VDD	V	
	IIL	输入泄漏电流 (2,3)				
D060		I/O 端口	—	±1	μA	VSS ≤ VPIN ≤ VDD, 引脚处于高阻态。
D061		MCLR	—	±5	μA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±5	μA	Vss ≤ VPIN ≤ VDD
	IPU	弱上拉电流				
D070	IPURB	PORTB 弱上拉电流	50	400	μA	VDD = 5V， VPIN = VSS

注 1： 在 RC 振荡器配置中， OSC1/CLKI 引脚为施密特触发器输入。在 RC 模式下，建议不要使用外部时钟驱动 PIC® 器件。

2： MCLR 引脚上的泄漏电流主要由施加在该引脚上的电平决定。规定电平为正常工作条件下的电平。在不同的输入电压下可测得更高的泄漏电流。

3： 负电流定义为引脚的拉电流。

4： 参数仅为特征值，未经测试。

22.3 直流规范： PIC18F1230/1330（工业级） PIC18LF1230/1330（工业级）（续）

直流规范			标准工作条件（除非另外声明） 工作温度 -40°C ≤ T_A ≤ +85°C（工业级）			
参数编号	符号	特性	最小值	最大值	单位	条件
	V_{OL}	输出低电压				
D080		I/O 端口	—	0.6	V	I_{OL} = 8.5 mA， V_{DD} = 4.5V， -40°C 至 +85°C
D083		OSC2/CLKO （RC、 RCIO、 EC 和 ECIO 模式）	—	0.6	V	I_{OL} = 1.6 mA， V_{DD} = 4.5V， -40°C 至 +85°C
	V_{OH}	输出高电压 [(3)]				
D090		I/O 端口	V_{DD} – 0.7	—	V	I_{OH} = -3.0 mA， V_{DD} = 4.5V， -40°C 至 +85°C
D092		OSC2/CLKO （RC、 RCIO、 EC 和 ECIO 模式）	V_{DD} – 0.7	—	V	I_{OH} = -1.3 mA， V_{DD} = 4.5V， -40°C 至 +85°C
		输出引脚上的容性负载规范				
D100[(4)]	C_{OSC2}	OSC2 引脚	—	15	pF	当外部时钟用于驱动 OSC1 时，处于 XT、 HS 或 LP 模式
D101	C_{IO}	所有 I/O 引脚和 OSC2 （在 RC 模式下）	—	50	pF	满足交流时序规范

注 1： 在 RC 振荡器配置中， OSC1/CLKI 引脚为施密特触发器输入。在 RC 模式下，建议不要使用外部时钟驱动 PIC® 器件。

2： $\overline{MCLR}$ 引脚上的泄漏电流主要由施加在该引脚上的电平决定。规定电平为正常工作条件下的电平。在不同的输入电压下可测得更高的泄漏电流。

3： 负电流定义为引脚的拉电流。

4： 参数仅为特征值，未经测试。



表 **22-1**： 存储器编程要求

直流规范			标准工作条件（除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C（工业级）				
参数编号	符号	特性	最小值	典型值 †	最大值	单位	条件
		数据 EEPROM 存储器					
D120	ED	字节的耐擦写能力	100K	1M	—	E/W	-40°C 至 +85°C
D121	VDRW	用于读 / 写的 VDD	VMIN	—	5.5	V	使用 EECON 读 / 写 VMIN = 最小工作电压
D122	TDEW	擦 / 写周期时间	—	4	—	ms	
D123	TRETD	保存时间	40	—	—	年	假如没有违反其他规范
D124	TREF	在刷新之前的总擦写周期数 (1)	1M	10M	—	E/W	-40°C 至 +85°C
D125	IDDP	编程期间的供电电流	—	10	—	mA	
		闪存程序存储器					
D130	EP	耐擦写能力	10K	100K	—	E/W	-40°C 至 +85°C
D131	VPR	用于读入的 VDD	VMIN	—	5.5	V	VMIN = 最小工作电压
D132B	VPEW	用于自定时写入的 VDD	VMIN	—	5.5	V	VMIN = 最小工作电压
D133A	TIW	自定时写周期	—	2	—	ms	
D134	TRETD	保存时间	40	100	—	年	假如没有违反其他规范
D135	IDDP	编程期间的供电电流	—	10	—	mA	

† 除非另外声明，“典型值”栏中的数据均为 5.0V、 25°C 下的值。这些参数仅供设计参考，未经测试。

注 1： 有关 EEPROM 耐擦写能力的更多细节，请参见第 **7.8** 节 **“使用数据 EEPROM”**。



表 **22-2**： 比较器规范

工作条件：除非另外声明，否则均为 3.0V < VDD < 5.5V， -40°C < TA < +85°C。							
参数编号	符号	特性	最小值	典型值	最大值	单位	备注
D300	VIOFF	输入失调电压	—	±5.0	±10	mV	
D301	VICM	输入共模电压	0	—	VDD – 1.5	V	
D302	CMRR	共模抑制比	55	—	—	dB	
300	TRESP	响应时间 [1]	—	150	400	ns	PIC18**F**XXXX
300A			—	150	600	ns	PIC18**LF**XXXX, VDD = 2.0V
301	TMC2OV	从比较器模式改变到输出有效的时间	—	—	10	µs	

注 **1**： 响应时间是在比较器的一个输入端电压为 (VDD – 1.5)/2，而另一个输入端从 VSS 跳变到 VDD 时测得的。

表 **22-3**： 参考电压规范

工作条件：除非另外声明，否则均为 3.0V < VDD < 5.5V， -40°C < TA < +85°C。							
参数编号	符号	特性	最小值	典型值	最大值	单位	备注
D310	VRES	分辨率	VDD/24	—	VDD/32	LSb	
D311	VRAA	绝对精度	—	—	1/2	LSb	
D312	VRUR	单位电阻值（R）	—	2k	—	Ω	
310	TSET	稳定时间 [1]	—	—	10	µs	

注 **1**： 稳定时间是在 CVRR = `1` 和 CVR3:CVR0 从 `0000` 跳变到 `1111` 时测得的。



图 **22-4**： 低电压检测特性

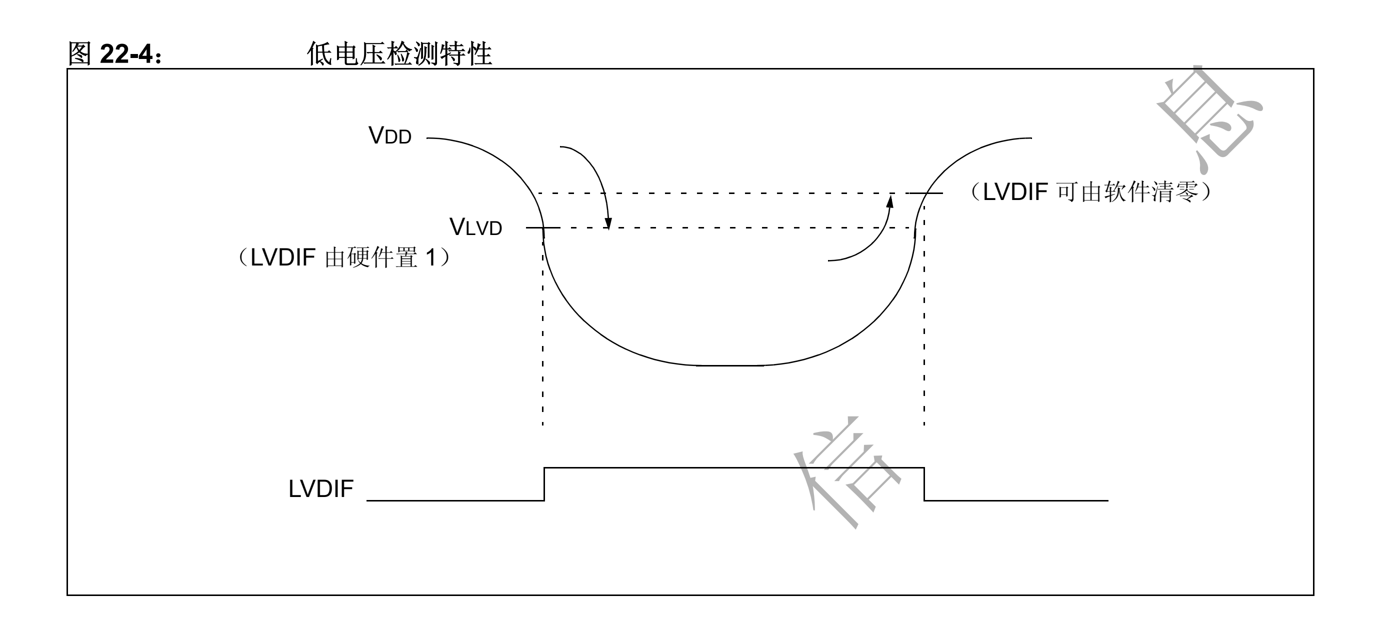


表 **22-4**： 低电压检测特性

标准工作条件 （除非另外声明）
工作温度 -40°C ≤ TA ≤ +85°C （工业级）

参数编号	符号	特性		最小值	典型值	最大值	单位	条件
D420		VDD 由高变低时的 LVD 电压	LVDL<3:0> = 0000	2.06	2.17	2.28	V	
			LVDL<3:0> = 0001	2.12	2.23	2.34	V	
			LVDL<3:0> = 0010	2.24	2.36	2.48	V	
			LVDL<3:0> = 0011	2.32	2.44	2.56	V	
			LVDL<3:0> = 0100	2.47	2.60	2.73	V	
			LVDL<3:0> = 0101	2.65	2.79	2.93	V	
			LVDL<3:0> = 0110	2.74	2.89	3.04	V	
			LVDL<3:0> = 0111	2.96	3.12	3.28	V	
			LVDL<3:0> = 1000	3.22	3.39	3.56	V	
			LVDL<3:0> = 1001	3.37	3.55	3.73	V	
			LVDL<3:0> = 1010	3.52	3.71	3.90	V	
			LVDL<3:0> = 1011	3.70	3.90	4.10	V	
			LVDL<3:0> = 1100	3.90	4.11	4.32	V	
			LVDL<3:0> = 1101	4.11	4.33	4.55	V	
			LVDL<3:0> = 1110	4.36	4.59	4.82	V	

22.4 交流（时序）特性



22.4.1 时序参数符号

可根据以下任一格式来创建时序参数符号：

1. TppS2ppS	3. TCC:ST （仅用于 I²C 规范）
2. TppS	4. Ts （仅用于 I²C 规范）

T			
F	频率	T	时间

小写字母（pp）及其含意：

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{\text{RD}}$
cs	$\overline{\text{CS}}$	rw	$\overline{\text{RD}}$ 或 $\overline{\text{WR}}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{\text{SS}}$
dt	数据输入	t0	T0CKI
io	I/O 端口	t1	T13CKI
mc	$\overline{\text{MCLR}}$	wr	$\overline{\text{WR}}$

大写字母及其含意：

S			
F	下降	P	周期
H	高	R	上升
I	无效（高阻态）	V	有效
L	低	Z	高阻态
仅用于 I²C 模式			
AA	输出通道	High	高
BUF	总线空闲	Low	低

TCC:ST（仅用于 I²C 规范）

CC			
HD	保持	SU	建立
ST			
DAT	保持数据输入	STO	停止条件
STA	启动条件		

22.4.2 时序条件

除非另外指明，表 22-5 中指定的温度和电压适用于所有时序规范。图 22-5 规定了时序规范的负载条件。

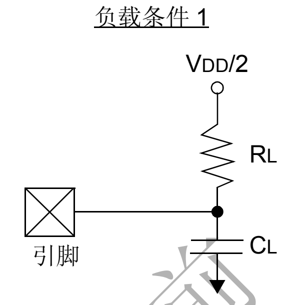
> **注：** 由于篇幅所限，本章节中通称的 “PIC18FXXXX” 和 “PIC18LFXXXX” 特指（而且仅指代）PIC18F1230/1330 和 PIC18LF1230/1330 系列器件。

表 22-5： 温度和电压规范－交流

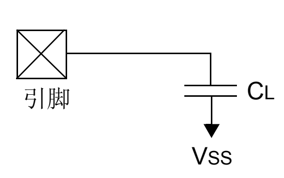
交流规范	**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） 直流规范第 **22.1** 节和第 **22.3** 节描述了工作电压 V_{DD} 的范围。 LF 器件仅在工业级温度下工作。

图 22-5： 器件时序规范的负载条件

负载条件 1



负载条件 2



R_L = 464Ω

C_L = 50 pF 适用于除 OSC2/CLKO 之外的所有引脚，也包括作为输出的 D 和 E 端口



22.4.3 时序图和规范

图 22-6： 外部时钟时序（除 PLL 之外的所有模式）

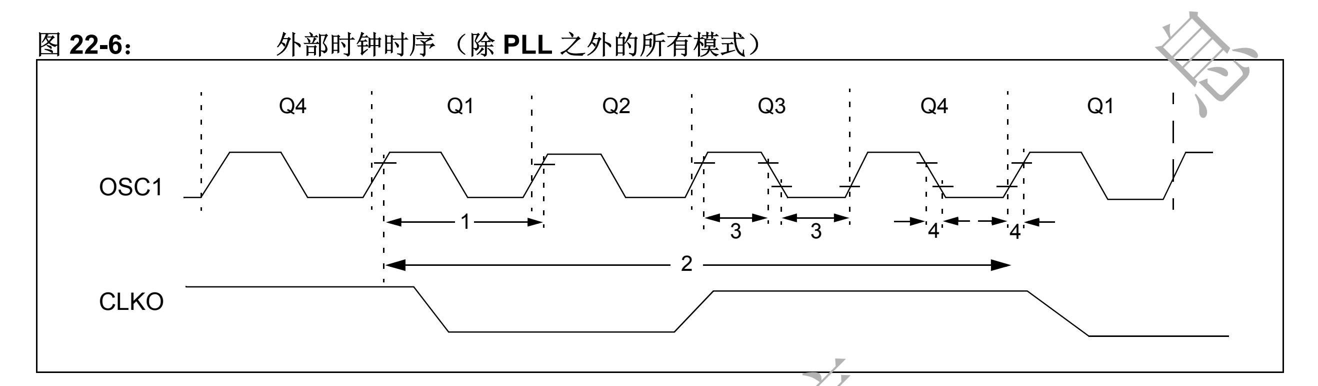


表 22-6： 外部时钟时序要求

参数编号	符号	特性	最小值	最大值	单位	条件
1A	FOSC	外部时钟 CLKI 频率 (1)	DC	1	MHz	XT 或 RC 振荡器模式
			DC	20	MHz	HS 振荡器模式
			DC	31.25	kHz	LP 振荡器模式
		振荡器频率 (1)	DC	4	MHz	RC 振荡器模式
			0.1	4	MHz	XT 振荡器模式
			4	20	MHz	HS 振荡器模式
			5	200	kHz	LP 振荡器模式
1	TOSC	外部时钟 CLKI 周期 (1)	1000	—	ns	XT 或 RC 振荡器模式
			50	—	ns	HS 振荡器模式
			32	—	μs	LP 振荡器模式
		振荡周期 (1)	250	—	ns	RC 振荡器模式
			250	1	μs	XT 振荡器模式
			100	250	ns	HS 振荡器模式
			50	250	ns	HS 振荡器模式
			5	—	μs	LP 振荡模式
2	TCY	指令周期时间 (1)	100	—	ns	TCY = 4/FOSC（工业级）
			160	—	ns	TCY = 4/FOSC（扩展级）
3	TOSL, TOSH	外部时钟输入（OSC1）的高电平或低电平时间	30	—	ns	XT 振荡器模式
			2.5	—	μs	LP 振荡器模式
			10	—	ns	HS 振荡器模式
4	TOSR, TOSF	外部时钟输入（OSC1）的上升或下降时间	—	20	ns	XT 振荡器模式
			—	50	ns	LP 振荡器模式
			—	7.5	ns	HS 振荡器模式

注 1： 对于除 PLL 的所有配置来说，指令周期时间（TCY）等于输入振荡器时基周期的 4 倍。所有值均为在特定的振荡器模式下，器件在标准工作条件下执行代码时获得的特征数据。超过规定值可能导致振荡器运行不稳定和 / 或电流消耗超出预期值。所有器件在测试 “最小值”时，都在 OSC1/CLKI 引脚连接了外部时钟。当使用了外部时钟输入时，所有器件的 “最大”周期时间限制为 “DC”（没有时钟）。

表 22-7： PLL 时钟时序规范 （VDD = 4.2V 至 5.5V）

参数编号	符号	特性	最小值	典型值 †	最大值	单位	条件
F10	FOSC	振荡器频率范围	4	—	10	MHz	仅 HS 模式
F11	FSYS	片上 VCO 系统频率	16	—	40	MHz	仅 HS 模式
F12	t_{rc}	PLL 起振时间 （锁定时间）	—	—	2	ms	
F13	ΔCLK	CLKO 稳定性 （抗抖动）	-2	—	+2	%	

† 除非另外声明，“典型值”栏中的数据均为 5V、 25°C 条件下的值。这些参数仅供设计参考，未经测试。

表 22-8： 交流规范： 内部 RC 精度
PIC18F1230/1330 （工业级）
PIC18LF1230/1330 （工业级）

PIC18LF1230/1330 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
PIC18F1230/1330 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	最小值	典型值	最大值	单位	条件	
	在频率为 8 MHz、 4 MHz、 2 MHz、 1 MHz、 500 kHz、 250 kHz 和 125 kHz 时的 INTOSC 精度 (1)						
	PIC18LF1230/1330	-2	+/-1	2	%	+25°C	VDD = 2.7-3.3V
		-5	—	5	%	-10°C 至 +85°C	VDD = 2.7-3.3V
		-10	+/-1	10	%	-40°C 至 +85°C	VDD = 2.7-3.3V
	PIC18F1230/1330	-2	+/-1	2	%	+25°C	VDD = 4.5-5.5V
		-5	—	5	%	-10°C 至 +85°C	VDD = 4.5-5.5V
		-10	+/-1	10	%	-40°C 至 +85°C	VDD = 4.5-5.5V
	频率为 31 kHz 时的 INTRC 精度 (2,3)						
	PIC18LF1230/1330	26.562	—	35.938	kHz	-40°C 至 +85°C	VDD = 2.7-3.3V
	PIC18F1230/1330	26.562	—	35.938	kHz	-40°C 至 +85°C	VDD = 4.5-5.5V

图注： 阴影行是为了增强表的可读性。
注 **1：** 频率校准温度为 25°C。 OSCTUNE 寄存器可用于补偿温度漂移。
2： 校准后的 INTRC 频率。
3： INTRC 频率随 VDD 的改变而改变。

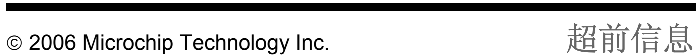
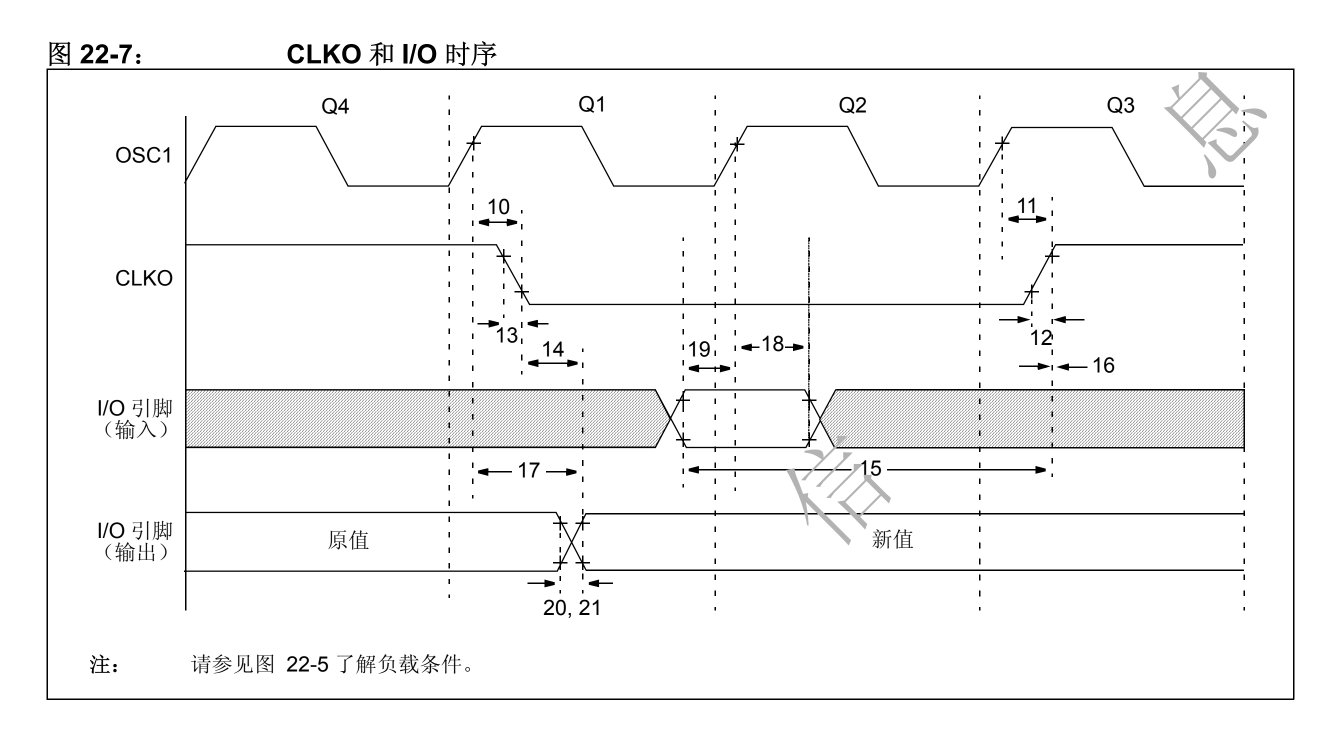


图 22-7： CLKO 和 I/O 时序



注： 请参见图 22-5 了解负载条件。

表 22-9： CLKO 和 I/O 时序要求

参数编号	符号	特性		最小值	典型值	最大值	单位	条件
10	TosH2ckL	OSC1 ↑ 到 CLKO ↓ 的时间		—	75	200	ns	（注 **1**）
11	TosH2ckH	OSC1 ↑ 到 CLKO ↑ 的时间		—	75	200	ns	（注 **1**）
12	TckR	CLKO 上升时间		—	35	100	ns	（注 **1**）
13	TckF	CLKO 下降时间		—	35	100	ns	（注 **1**）
14	TckL2ioV	CLKO ↓ 到端口输出有效的时间		—	—	0.5 TCY + 20	ns	（注 **1**）
15	TioV2ckH	CLKO ↑ 前端口输入有效时间		0.25 TCY +25	—	—	ns	（注 **1**）
16	TckH2ioI	在 CLKO ↑ 后端口输入保持时间		0	—	—	ns	（注 **1**）
17	TosH2ioV	OSC1 ↑ （Q1 周期）到端口输出有效的时间		—	50	150	ns	
18	TosH2ioI	OSC1 ↑ （Q2 周期）到端口输入无效的时间 （I/O 输入保持时间）	PIC18**F**XXXX	100	—	—	ns	
18A			PIC18**LF**XXXX	200	—	—	ns	VDD = 2.0V
19	TioV2osH	端口输入有效到 OSC1 ↑ （I/O 输入建立时间）		0	—	—	ns	
20	TioR	端口输出上升时间	PIC18**F**XXXX	—	10	25	ns	
20A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
21	TioF	端口输出下降时间	PIC18**F**XXXX	—	10	25	ns	
21A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
22†	TINP	INTx 引脚高电平或低电平时间		TCY	—	—	ns	
23†	TRBP	RB7:RB4 电平变化中断 INTx 高电平或低电平时间		TCY	—	—	ns	

† 这些参数是与内部时钟边沿无关的异步事件。

注 **1**： 测量是在 RC 模式下进行的，其中 CLKO 输出为 4 x TOSC。

图 22-8： 复位、看门狗定时器、振荡器起振定时器和上电延时定时器时序

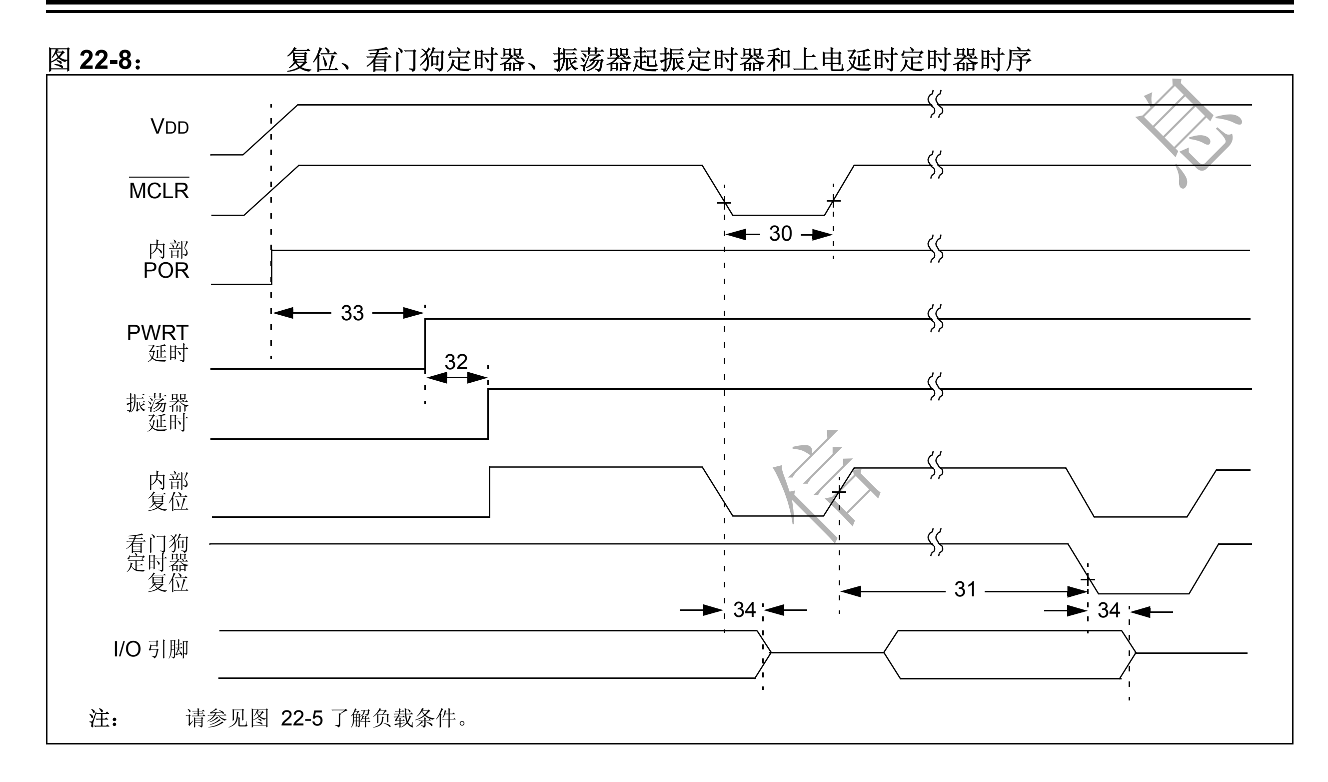


图 22-9： 欠压复位时序

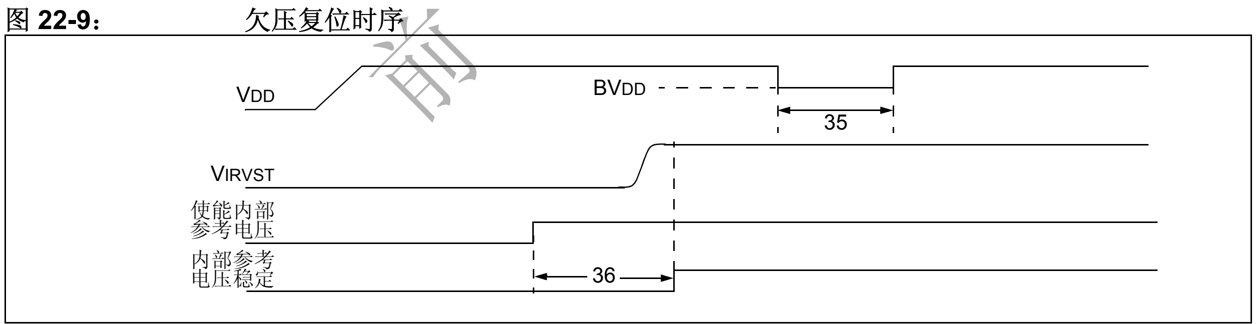
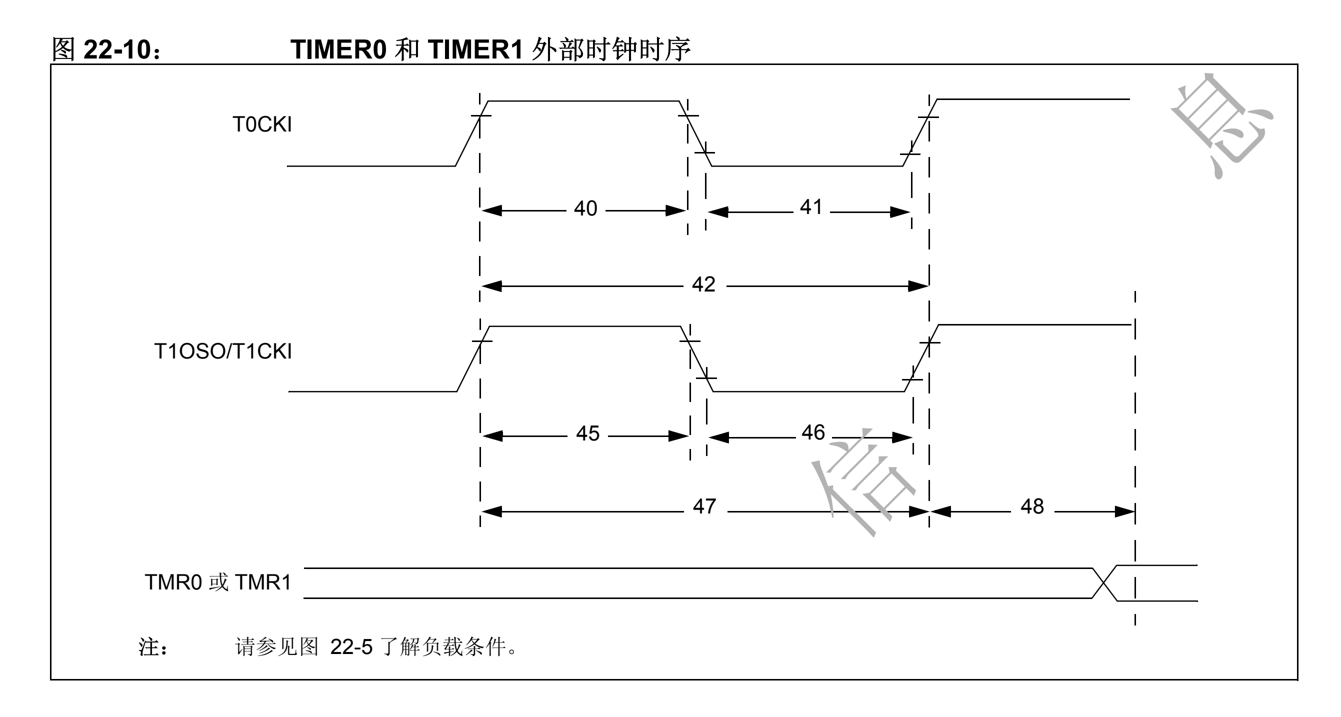


表 22-10： 复位、看门狗定时器、振荡器起振定时器、上电延时定时器和欠压复位要求

参数编号	符号	特性	最小值	典型值	最大值	单位	条件
30	TmcL	MCLR 脉冲宽度 （低电平）	2	—	—	μs	
31	TWDT	看门狗定时器超时周期 （无后分频器）	3.4	4.0	4.6	ms	
32	TOST	振荡器起振定时器周期	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 周期
33	TPWRT	上电延时定时器周期	55.6	65.5	75	ms	
34	TIOZ	自 MCLR 低电平或看门狗定时器复位起 I/O 处于高阻态的时间	—	2	—	μs	
35	TBOR	欠压复位脉冲宽度	200	—	—	μs	VDD ≤ BVDD （见 D005）
36	TIRVST	内部参考电压稳定时间	—	20	50	μs	
37	TLVD	低电压检测脉冲宽度	200	—	—	μs	VDD ≤ VLVD
38	TCSD	CPU 的启动时间	—	10	—	μs	
39	TIOBST	INTOSC 电路稳定时间	—	1	—	μs	

图 22-10： TIMER0 和 TIMER1 外部时钟时序



注： 请参见图 22-5 了解负载条件。

表 22-11： TIMER0 和 TIMER1 外部时钟要求

参数编号	符号	特性			最小值	最大值	单位	条件
40	Tt0H	T0CKI 高电平脉冲宽度		无预分频器	0.5 TCY + 20	—	ns	
				有预分频器	10	—	ns	
41	Tt0L	T0CKI 低电平脉冲宽度		无预分频器	0.5 TCY + 20	—	ns	
				有预分频器	10	—	ns	
42	Tt0P	T0CKI 周期		无预分频器	TCY +10	—	ns	
				有预分频器	取较大值： 20 ns 或 (TCY + 40)/N	—	ns	N = 预分频值 （1、2、 4、 ... 或 256）
45	Tt1H	T1CKI 高电平时间	同步，无预分频器		0.5 TCY + 20	—	ns	
			同步， 有预分频器	PIC18**F**XXXX	10	—	ns	
				PIC18**LF**XXXX	25	—	ns	VDD = 2.0V
			异步	PIC18**F**XXXX	30	—	ns	
				PIC18**LF**XXXX	50	—	ns	VDD = 2.0V
46	Tt1L	T1CKI 低电平时间	同步，无预分频器		0.5 TCY +5	—	ns	
			同步， 有预分频器	PIC18**F**XXXX	10	—	ns	
				PIC18**LF**XXXX	25	—	ns	VDD = 2.0V
			异步	PIC18**F**XXXX	30	—	ns	
				PIC18**LF**XXXX	50	—	ns	VDD = 2.0V
47	Tt1P	T1CKI 输入周期	同步		取较大值： 20 ns 或 (TCY + 40)/N	—	ns	N = 预分频值 (1、 2、 4 和 8）
			异步		60	—	ns	
	Ft1	T1CKI 振荡器输入频率范围			DC	50	kHz	
48	Tcke2tmrl	从外部 T1CKI 时钟边沿到定时器递增的延时			2 TOSC	7 TOSC	—	

图 **22-11**： **EUSART** 同步发送 （主控 / 从动）时序

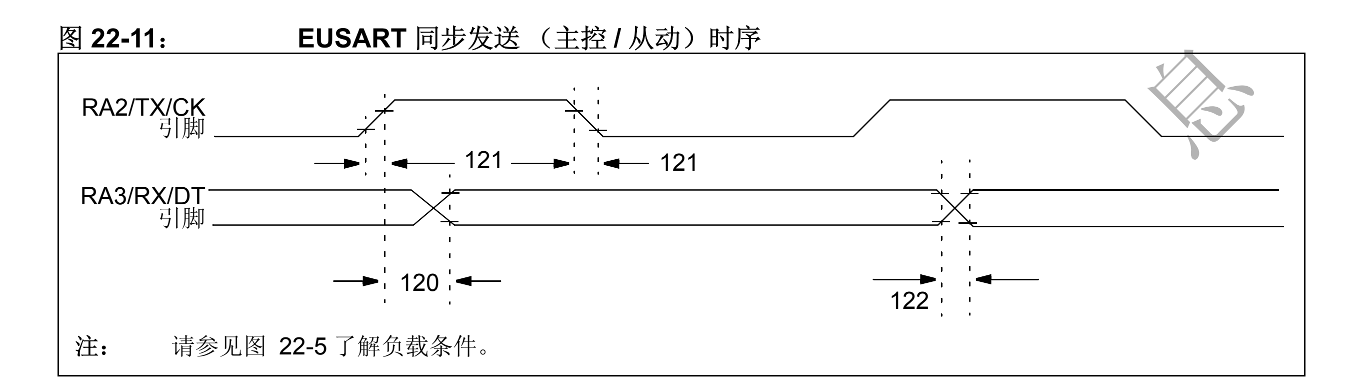


表 **22-12**： **EUSART** 同步发送要求

参数编号	符号	特性		最小值	最大值	单位	条件
120	TckH2dtV	SYNC XMIT （主控和从动） 从时钟高电平到数据输出有效的时间	PIC18**F**XXXX	—	40	ns	
			PIC18**LF**XXXX	—	100	ns	VDD = 2.0V
121	Tckrf	时钟输出信号的上升时间和下降时间（主控模式）	PIC18**F**XXXX	—	20	ns	
			PIC18**LF**XXXX	—	50	ns	VDD = 2.0V
122	Tdtrf	数据输出信号的上升时间和下降时间	PIC18**F**XXXX	—	20	ns	
			PIC18**LF**XXXX	—	50	ns	VDD = 2.0V



图 **22-12**： **EUSART** 同步接收 （主控 / 从动）时序

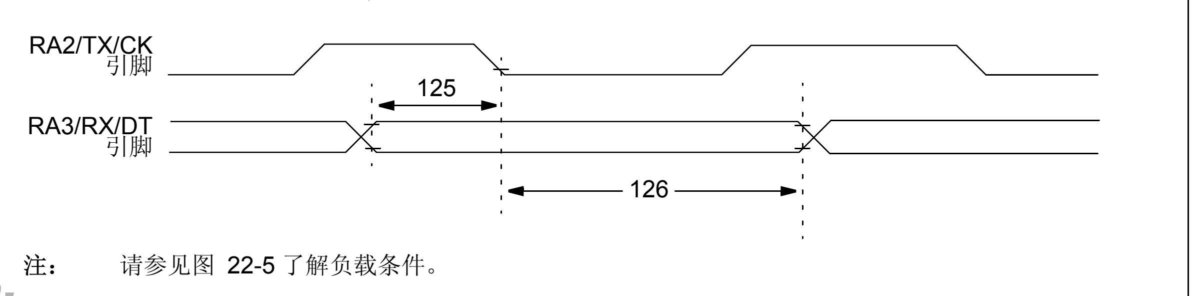


表 **22-13**： **EUSART** 同步接收要求

参数编号	符号	特性	最小值	最大值	单位	条件
125	TdtV2ckl	同步接收 （主控和从动） 在 CK ↓之前数据的保持时间 （DT 保持时间）	10	—	ns	
126	TckL2dtl	在 CK ↓之后数据的保持时间 （DT 保持时间）	15	—	ns	

表 22-14： **A/D 转换器特性：** **PIC18F1230/1330**（工业级）
PIC18LF1230/1330（工业级）

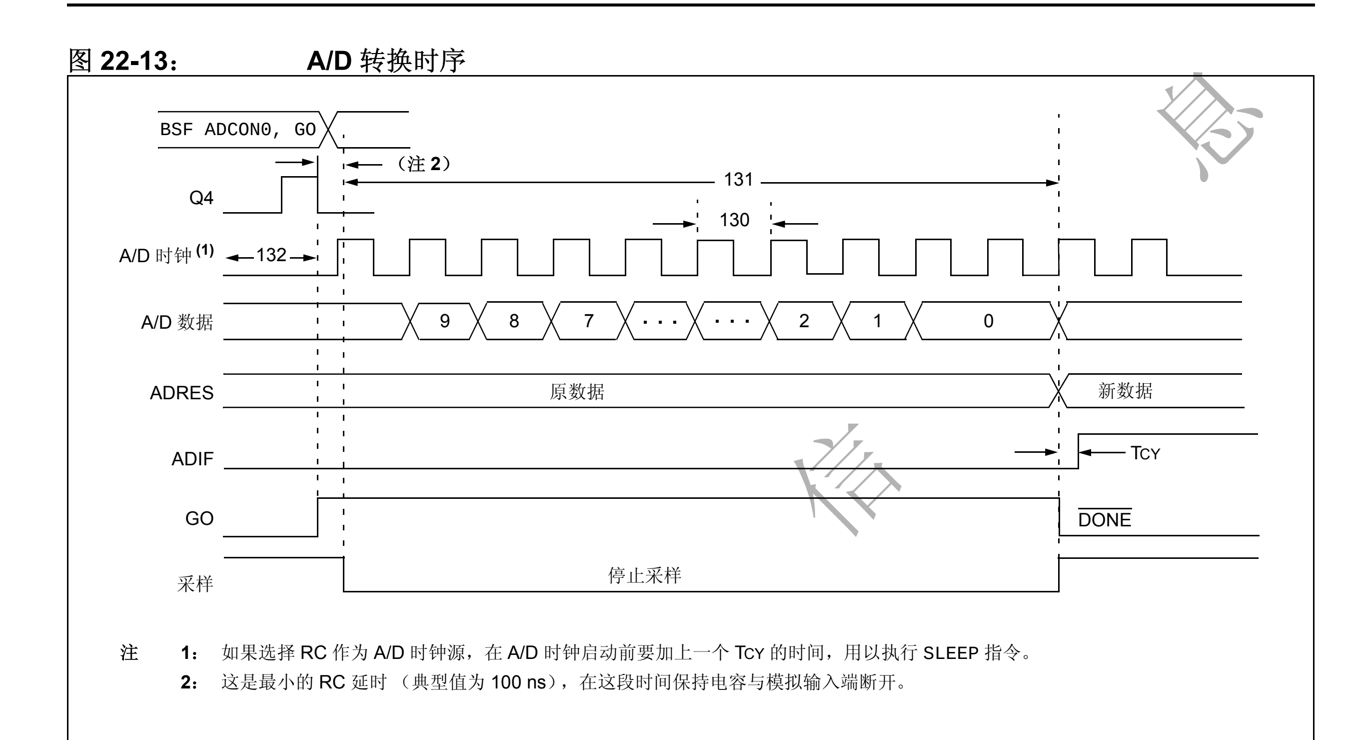
参数编号	符号	特性	最小值	典型值	最大值	单位	条件
A01	N_R	分辨率	—	—	10	bit	$\Delta V_{REF} \geq 3.0V$
A03	E_{IL}	积分线性误差	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A04	E_{DL}	微分线性误差	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A06	E_{OFF}	偏移误差	—	—	$< \pm 1.5$	LSb	$\Delta V_{REF} \geq 3.0V$
A07	E_{GN}	增益误差	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A10	—	单调性	保证[1]			—	$V_{SS} \leq V_{AIN} \leq V_{REF}$
A20	ΔV_{REF}	参考电压范围 （$V_{REF+} - V_{SS}$）	1.8 3	— —	— —	V V	$V_{DD} < 3.0V$ $V_{DD} \geq 3.0V$
A21	V_{REF+}	正向参考电压	V_{SS}	—	V_{REF+}	V	
A25	V_{AIN}	模拟输入电压	V_{SS}	—	V_{REF+}	V	
A30	Z_{AIN}	模拟电源阻抗推荐值	—	—	2.5	kΩ	
A50	I_{REF}	V_{REF+} 输入电流[2]	—	—	5 150	μA μA	在采集 V_{AIN} 期间。 在 A/D 转换期间。

注 **1：** A/D 转换结果不会因输入电压的增加而减小，并且不会丢失编码。
2： V_{REF+} 电流来自选作 V_{REF+} 源的 RA4/T0CKI/AN2/V_{REF+} 引脚或 V_{DD}。





图 22-13： A/D 转换时序



注 1： 如果选择 RC 作为 A/D 时钟源，在 A/D 时钟启动前要加上一个 TCY 的时间，用以执行 SLEEP 指令。

2： 这是最小的 RC 延时 （典型值为 100 ns），在这段时间保持电容与模拟输入端断开。

表 22-15： A/D 转换要求

参数编号	符号	特性		最小值	最大值	单位	条件
130	TAD	A/D 时钟周期	PIC18**F**XXXX	0.7	25.0(1)	μs	基于 TOSC， VREF ≥3.0V
			PIC18**LF**XXXX	1.4	25.0(1)	μs	VDD = 2.0V, 基于 TOSC， VREF 满量程
			PIC18**F**XXXX	TBD	1	μs	A/D RC 模式
			PIC18**LF**XXXX	TBD	3	μs	VDD = 2.0V， A/D RC 模式
131	TCNV	转换时间（不包括采集时间） (2)		11	12	TAD	
132	TACQ	采集时间 (3)		1.4 TBD	— —	μs μs	-40°C 至 +85°C 0°C 至 +85°C
135	TSWC	转换 → 采样的切换时间		—	（注 4）		
TBD	TDIS	电容放电时间		0.2	—	μs	

图注： TBD = 待定。

注 1： A/D 时钟周期取决于器件频率和 TAD 时钟分频比。

2： 可在下一个 TCY 周期读取 ADRES 寄存器。

3： 转换完成后当电压满量程变化时 （VDD 至 VSS，或 VSS 至 VDD），保持电容采样一个 “新”输入电压所需的时间。在输入通道上的源阻抗 （RS）为 50Ω。

4： 在器件时钟的下一个周期上。

注：

23.0 DC 和 AC 特性图表

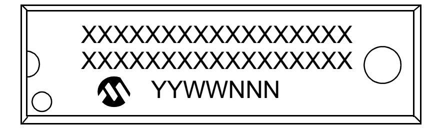
当前没有图表。

注：

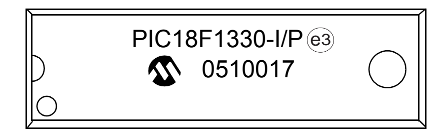
24.0 封装信息

24.1 封装标识信息

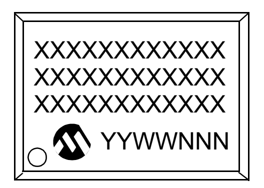
18 引脚 PDIP



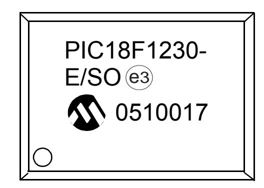
示例



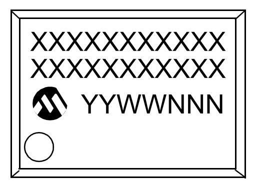
18 引脚 SOIC



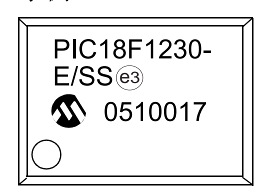
示例



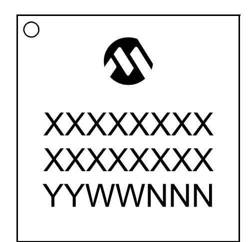
20 引脚 SSOP



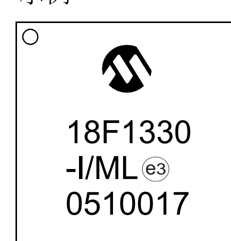
示例



28 引脚 QFN



示例



图注：		
	XX...X	客户信息
	Y	年份代码（日历年的最后一位数字）
	YY	年份代码（日历年的最后两位数字）
	WW	星期代码（一月一日的星期代码为 “01”）
	NNN	以字母数字排序的追踪代码
	e3	雾锡（Matte Tin， Sn）的 JEDEC 无铅标志
	*	表示无铅封装。 JEDEC 无铅标志（e3）标示于此种封装的外包装上。

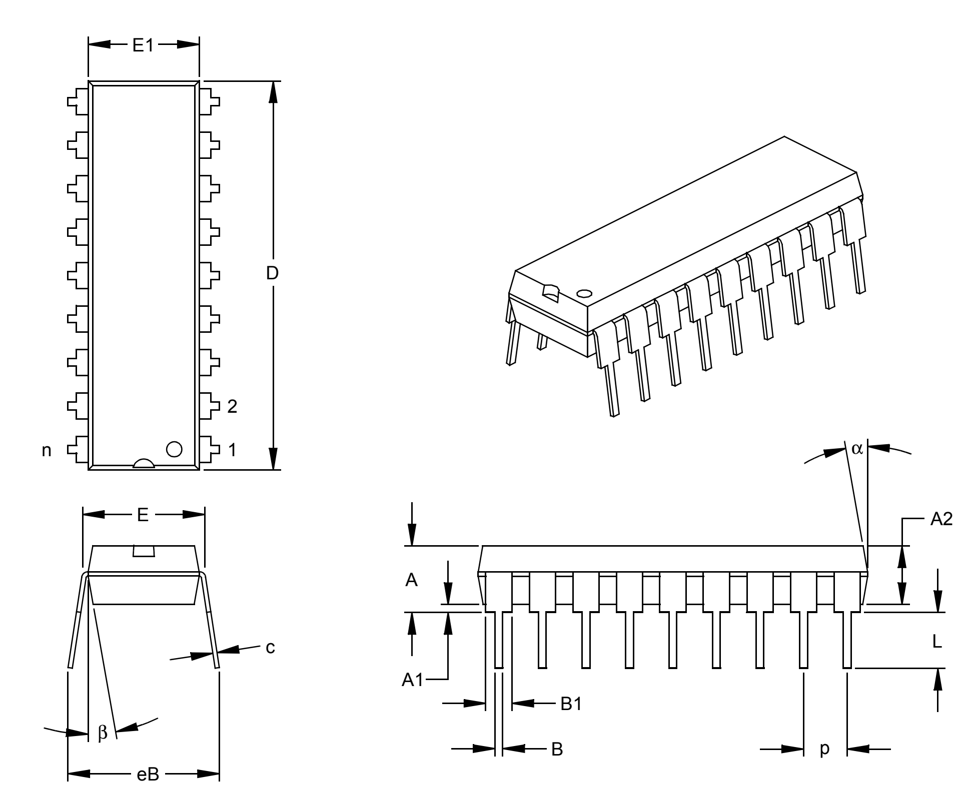
注： Microchip 元器件编号如果无法在同一行内完整标注，将换行标出，因此会限制表示客户信息的字符数。

24.2 封装详细信息

以下部分将介绍各种封装的技术细节。

18 引脚塑封双列直插式封装 （P）——主体 300 mil （PDIP）

注：	最新封装图请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



	单位	英寸 *			毫米		
	尺寸范围	最小	正常	最大	最小	正常	最大
引脚数	n		18			18	
引脚间距	p		.100			2.54	
顶端到固定面高度	A	.140	.155	.170	3.56	3.94	4.32
塑模封装厚度	A2	.115	.130	.145	2.92	3.30	3.68
塑模底面到固定面高度	A1	.015			0.38		
肩到肩宽度	E	.300	.313	.325	7.62	7.94	8.26
塑模封装宽度	E1	.240	.250	.260	6.10	6.35	6.60
总长度	D	.890	.898	.905	22.61	22.80	22.99
引脚尖到固定面高度	L	.125	.130	.135	3.18	3.30	3.43
引脚厚度	c	.008	.012	.015	0.20	0.29	0.38
引脚上部宽度	B1	.045	.058	.070	1.14	1.46	1.78
引脚下部宽度	B	.014	.018	.022	0.36	0.46	0.56
总排列间距 §	eB	.310	.370	.430	7.87	9.40	10.92
塑模顶部锥度	α	5	10	15	5	10	15
塑模底部锥度	β	5	10	15	5	10	15

* 控制参数

§ 重要特性

注：

尺寸 D 和 E1 不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过 0.010 英寸（0.254 毫米）。

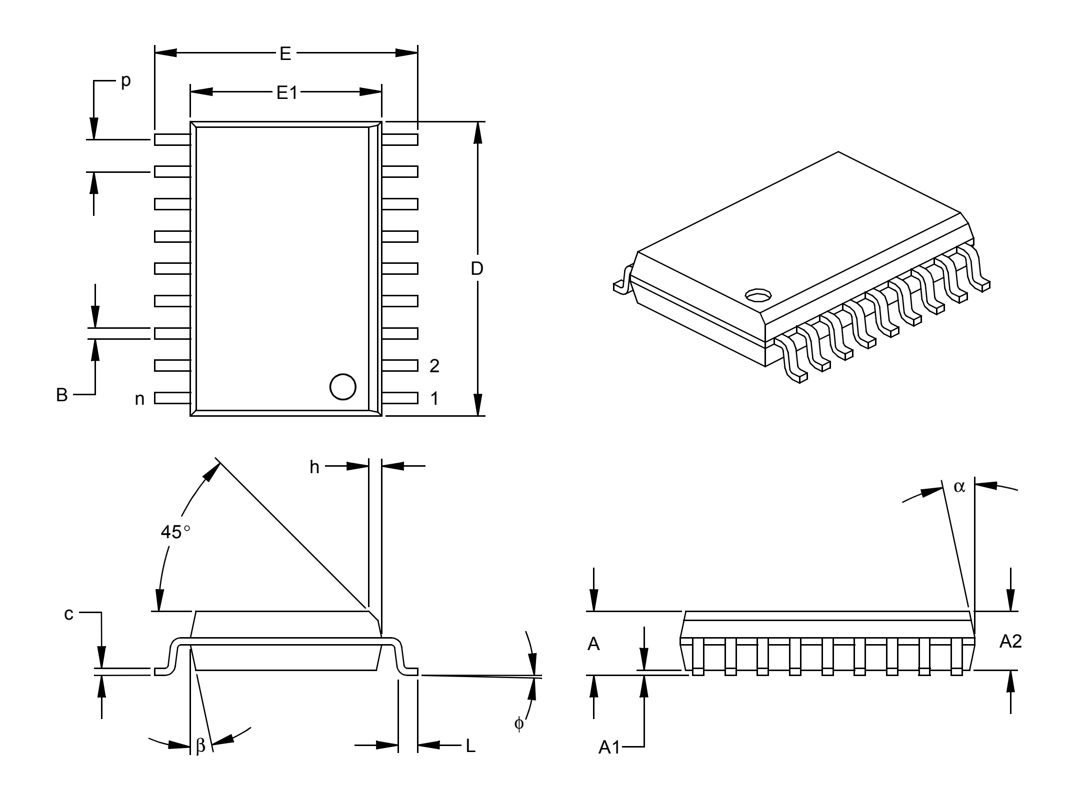
等同于 JEDEC 号：MS-001

图号：C04-007

超前信息

18 引脚宽型塑封小外形封装 （SO）——主体 300 mil （SOIC）

注：	最新封装图请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。

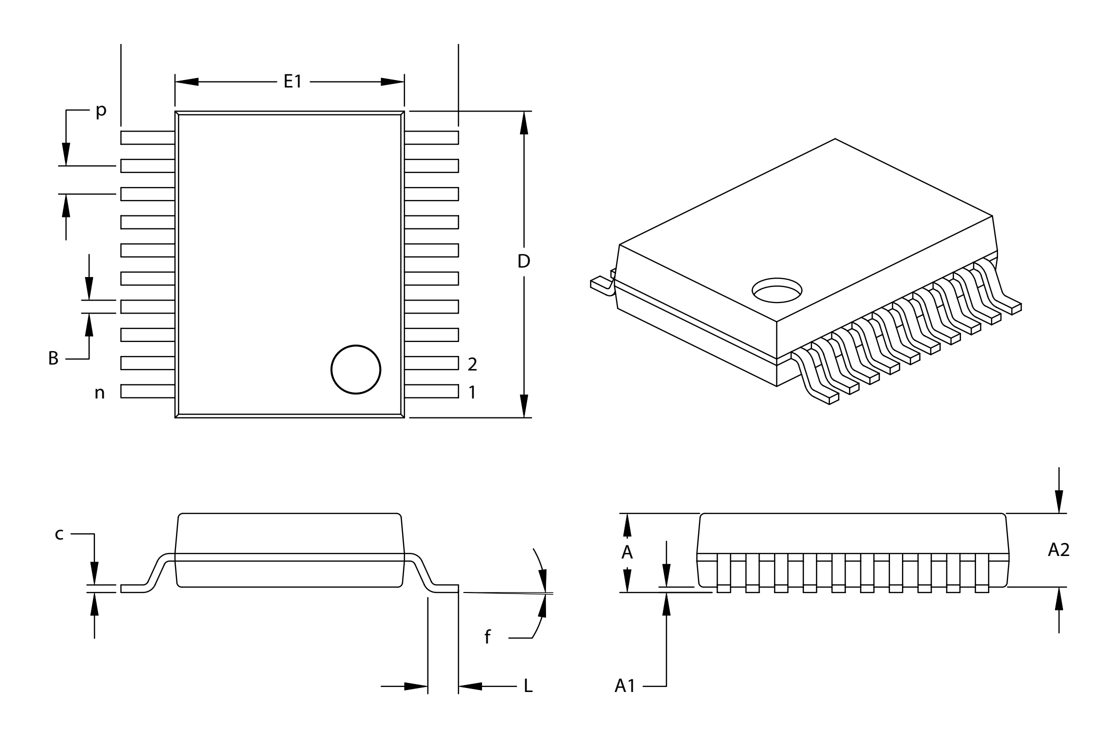


单位		英寸 *			毫米		
尺寸范围		最小	正常	最大	最小	正常	最大
引脚数	n		18			18	
引脚间距	p		.050			1.27	
总高度	A	.093	.099	.104	2.36	2.50	2.64
塑模封装厚度	A2	.088	.091	.094	2.24	2.31	2.39
悬空间隙 §	A1	.004	.008	.012	0.10	0.20	0.30
总宽度	E	.394	.407	.420	10.01	10.34	10.67
塑模封装宽度	E1	.291	.295	.299	7.39	7.49	7.59
总长度	D	.446	.454	.462	11.33	11.53	11.73
斜面投影距离	h	.010	.020	.029	0.25	0.50	0.74
底脚长度	L	.016	.033	.050	0.41	0.84	1.27
底脚倾角	φ	0	4	8	0	4	8
引脚厚度	c	.009	.011	.012	0.23	0.27	0.30
引脚宽度	B	.014	.017	.020	0.36	0.42	0.51
塑模顶部锥度	α	0	12	15	0	12	15
塑模底部锥度	β	0	12	15	0	12	15

* 控制参数
§ 重要特性
注：
尺寸 D 和 E1 不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过 0.010 英寸（0.254 毫米）
等同于 JEDEC 号：MS-013
图号：C04-051

20 引脚塑封缩小外形封装 （SS）——主体 209mil， 5.30 mm （SSOP）

注：	最新封装图请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



单位		英寸			毫米*		
尺寸范围		最小	正常	最大	最小	正常	最大
引脚数	n		20			20	
引脚间距	p		.026			0.65	
总高度	A	-	-	.079	-	-	2.00
塑模封装厚度	A2	.065	.069	.073	1.65	1.75	1.85
悬空间隙	A1	.002	-	-	0.05	-	-
总宽度	E	.291	.307	.323	7.40	7.80	8.20
塑模封装宽度	E1	.197	.209	.220	5.00	5.30	5.60
总长度	D	.272	.283	.295	6.90	7.20	7.50
底脚长度	L	.022	.030	.037	0.55	0.75	0.95
引脚厚度	c	.004	-	.010	0.09	-	0.25
底脚倾斜角	f	0°	4°	8°	0°	4°	8°
引脚宽度	B	.009	-	.015	0.22	-	0.38

*控制参数

注：

尺寸D和E1不包括塑模毛边或突起。塑模每测的毛边或突起不得超过 0.010英寸（0.254mm）。

等同于JEDEC号：MO-150

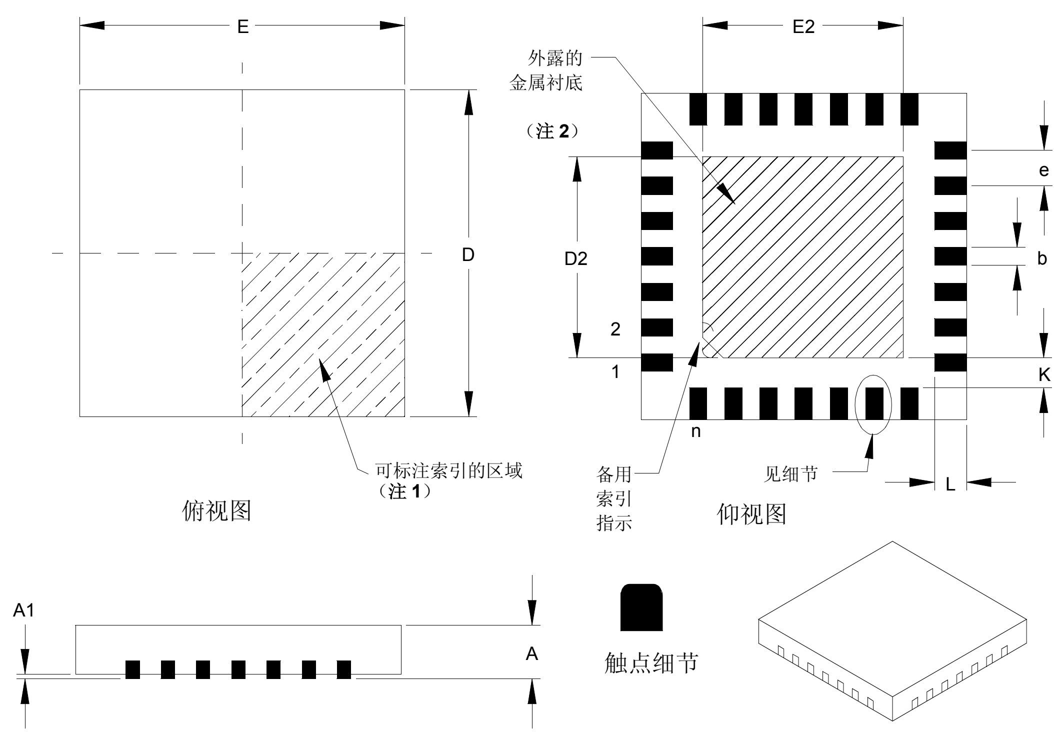
图号 C04-072

修订于8-27-04

超前信息

28 引脚塑封正方扁平无脚封装 （ML）主体 6x6 mm （QFN）——触点长度为 0.55 mm （Saw Singulated）

注：	最新封装图请至 http://www.microchip.com/packaging 查看 Microchip 封装规范。



单位		英寸			毫米 *		
尺寸范围		最小	正常	最大	最小	正常	最大
引脚数	n		28			28	
引脚间距	e		.026 BSC			.65 BSC	
总高度	A	.031	.035	.039	0.80	0.90	1.00
悬空间隙	A1	.000	.001	.002	0.00	0.02	0.05
触点厚度	A3		.008 REF.			.20 REF.	
总宽度	E	.232	.236	.240	5.90	6.00	6.10
外露金属衬底宽度	E2	.153	.167	.169	3.89	4.24	4.29
总长度	D	.232	.236	.240	5.90	6.00	6.10
外露金属衬底长度	D2	.153	.167	.169	3.89	4.24	4.29
触点宽度	β	.009	.011	.013	0.23	0.28	0.33
触点长度 §	L	.018	.022	.024	0.45	0.55	0.65
触点到外露金属衬底的距离 §	K	.008	–	–	0.20	–	–

* 控制参数
§ 重要特性

注：

1. 引脚 1 的索引外观可以变化，但必须位于阴影区域内。
2. 外露金属衬底尺寸随管芯叶片大小而变化。

BSC：基本尺寸。理论值，不带公差。
请参见 ASME Y14.5M
REF：参考尺寸，通常不带公差，仅作为参考信息。
请参见 ASME Y14.5M
等同于 JEDEC 号：MO-220
图号：C04-105

修订于 09-12-05

注：

附录 A： 版本历史

版本 A （2005 年 11 月）

PIC18F1230/1330 器件数据手册的最初版本。

版本 B （2006 年 2 月）

更新了有关数据存储区的信息，并且为计算 PCPWM 占空比添加了注解。

附录 B： 器件差异

表 B-1 为本数据手册中所列器件之间的差异。

表 B-1： 器件差异

特性	PIC18F1230	PIC18F1330
程序存储器 （字节数）	4096	8192
程序存储器 （指令条数）	2048	4096
封装	18 引脚 PDIP 18 引脚 SOIC 20 引脚 SSOP 28 引脚 QFN	18 引脚 PDIP 18 引脚 SOIC 20 引脚 SSOP 28 引脚 QFN

附录 C： 转换注意事项

本附录讨论了器件从老版本升级到数据手册中所列版本时的注意事项。这些变化通常是由于采用的工艺技术不同而引起的，正如从 PIC16C74A 至 PIC16C74B 的升级。

不适用

附录 D： 从基本型器件移植到增强型器件

本节讨论如何从基本型器件（即 PIC16C5X）移植到增强型 MCU 器件 （即 PIC18FXXX）。

下表列出了对 PIC16C5X 系列单片机所做的修改：

当前没有数据

附录 E： 从中档器件移植到增强型器件

在 AN716 “*Migrating Designs from PIC16C74A/74B to PIC18F442*” 中详细讨论了中档 MCU 器件（即 PIC16CXXX）与增强型器件（即 PIC18FXXX）之间的差异。虽然所讨论的内容是针对特定器件的，但是适用于从中档器件移植至增强型器件的所有情况。

上述应用笔记的文献编号为 DS00716。

附录 F： 从高档器件移植到增强型器件

在 AN726 “*PIC17CXXX to PIC18CXXX Migration*” 中详细讨论了从高档 MCU 器件（即 PIC17CXXX）移植到增强型器件（即 PIC18FXXX）的步骤以及两者之间的差异。

上述应用笔记的文献编号为 DS00726。

注：

索引

A

B

C

D

J

K

超前信息

 超前信息

注：

MICROCHIP 网站

Microchip 网站 （www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的因特网浏览器即可访问。网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和样本程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及存档软件
- **一般技术支持**——常见问题 （FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 www.microchip.com，点击 “变更通知客户 （Customer Change Notification）”服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师 （FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师 （FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http：//support.microchip.com 获得网上技术支持。

读者反馈表

我们努力为您提供最佳文档，以确保您能够成功使用 Microchip 产品。如果您对文档的组织、条理性、主题及其他有助于提高文档质量的方面有任何意见或建议，请填写本反馈表并传真给我公司 TRC 经理，传真号码为 86-21-5407-5066。

请填写以下信息，并从下面各方面提出您对本文档的意见。

致: TRC 经理　　　　总页数 ________

关于: 读者反馈

发自: 姓名 ________________________________

公司 ________________________________

地址 ________________________________

国家 / 省份 / 城市 / 邮编 ________________________________

电话: (______) ________________　　　　传真: (______) ________________

应用（选填）:

您希望收到回复吗？是____ 否____

器件: PIC18F1230/1330　　　　文献编号: DS39758B_CN

问题

1. 本文档中哪些部分最有特色？

____

____

2. 本文档是否满足了您的软硬件开发要求？如何满足的？

____

____

3. 您认为本文档的组织结构便于理解吗？如果不便于理解，那么问题何在？

____

____

4. 您认为本文档应该添加哪些内容以改善其结构和主题？

____

____

5. 您认为本文档中可以删减哪些内容，而又不会影响整体使用效果？

____

____

6. 本文档中是否存在错误或误导信息？如果存在，请指出是什么信息及其具体页数。

____

____

7. 您认为本文档还有哪些方面有待改进？

____

____

PIC18F1230/1330 产品标识体系

欲订货或获取价格、交货等信息，请与我公司生产厂或销售办事处联系。

器件编号	X	/XX	XXX
器件	温度范围	封装	模板

器件 PIC18F1230/1330(1)
PIC18F1230/1330T(2)
VDD 范围为 4.2V 至 5.5V
PIC18LF1230/1330(1)
PIC18LF1230/1330T(2)
VDD 范围为 2.0V 至 5.5V

温度范围
I = -40°C 至 +85°C （工业级）
E = -40°C 至 +125°C （扩展级）

封装 (3)
SO = 塑封小外形封装 （SOIC）
SO = 塑封缩小外形封装 （SOIC）
P = 塑封双列直插式封装 （PDIP）
ML = 塑封正方扁平无脚封装 （QFN）

模板 QTP、SQTP、代码或特殊要求
（其他情况空白）

示例：

a) PIC18LF1330-I/P 301 = 工业级温度、PDIP 封装、扩展级 VDD 范围和 QTP 模板 #301。
b) PIC18LF1230-I/SO = 工业级温度、SOIC 封装和扩展级 VDD 范围。

注 1： F = 标准电压范围
LF = 宽电压范围
2： T = 仅卷带式 QFN 封装



全球销售及服务网点

美洲

公司总部 **Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 **Atlanta**
Alpharetta, GA
Tel: 1-770-640-0034
Fax: 1-770-640-0307

波士顿 **Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 **Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 **Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 **Detroit**
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 **Kokomo**
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 **Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 **Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 **Toronto**
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 **Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Habour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 **Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 **India - Bangalore**
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

印度 **India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 **India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 **Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 **Korea - Gumi**
Tel: 82-54-473-4301
Fax: 82-54-473-4302

韩国 **Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 **Malaysia - Penang**
Tel: 60-4-646-8870
Fax: 60-4-646-5086

菲律宾 **Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 **Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 **Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 **Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 **Denmark-Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 **France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 **Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 **Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 **Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 **Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 **UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/19/06